VIBRATION SENSING FOR ENGINE CONDITION MONITORING AND PREDICTIVE MAINTENANCE

Masinde Masinde Mtesigwa 41969
Master's Thesis in Computer Engineering
Supervisors: Jerker Björkqvist, Andreas Lundell
Faculty of Science and Engineering
Åbo Akademi University
2020

Abstract

Edge computing and Edge analytics which can also be called fog computing send only the critical data to the cloud after filtering, aggregation and analysis at the edge. The idea is to bring computing, storage and analytics closer to the source of the data. This is because of the problem which IoT cannot solve. The edge computing helps Industry 4.0 to expand to the areas where IoT cannot reach and also solves the high costs of bandwidth. The latency problem due to the architecture gives birth to edge. Cloud is the ultimate destination where all the edge devices will send the filtered, aggregated data for analysis and building models.

Edge computing is one of the emerging paradigms which have become a feasible supplementary solution under the cloud architecture by increasing computing and storage resources available at the network edge. The edge solves the problems of latency of sending to the cloud and the performance burden under IoT. [1]. Edge computing has the potential to deliver low-latency, bandwidth-efficient and resilient services to IoT devices via adopting platforms that provide intermediate layers of computation, networking and storage on the edge.[2]

The envisioned edge-driven IoT environment consists of three layers: IoT devices, edge layer, and cloud. The main layer, edge, plays the crucial role of bridging and interfacing the central cloud with IoT.[3] The edge element in this thesis is Adlink Fanless Embedded Computer which provides computing capability for data analysis, storage, and networking resources to the applications deployed across IoT devices and cloud. Edge Computing is a highly virtualised platform which has computing, storage, and networking services between end devices and traditional Cloud Computing Data Centres. [4]

Today, an increasing part of added value for new technical solutions comes from digitalisation and advanced automation. Through the use of big data and cloud analytics, machines can be made more reliable, more energy-efficient and the operation can be optimised. In this thesis, the target is to transfer the capabilities of big data and cloud processing to the edge, enabling real-time safety-critical operation, regardless of communication availability and at the same time minimising data transfer costs. The developed solution and methods are utilised for the needs of machine diagnostic and analytics.

This thesis presents edge analytics focusing on predictive maintenance which happens to be the critical component of well-functioning or smart industry. The aim of predictive maintenance is monitoring production equipment in the industry during operation to detect early warning signs of potential failures, which may lead to a shutdown of the industry for maintenance reasons. Edge computing in this thesis aims to find a possible solution through vibration monitoring using MEMS sensors. Further, the project aims to utilise information from the designing phase of equipment and physical context (Digital twin), feedback of expert personnel (human-in-the-loop) and capacity of cloud environment in training, initialisation and parametrisation, are reached for so that optimal, up-to-date analytics can be guaranteed. The developed solutions and methods will be verified, validated using demonstrators (ship engine, forest machine, shipyard crane) and data collected from these systems.

Keywords: IoT, Edge Computing, machine learning, anomaly detection

Contents

Introduction

2.1 Overview

Industry 4.0 is snowballing with new demands growing, which has resulted in generation of massive data. The gigantic data has costs to Industrial 4.0, such as latency in sending data to the cloud for analysis, small geographical expansion to due lack of internet, and security vulnerability. When using the cloud, data generated goes to the cloud data centre before processing. However, considering the amount of data that today is being intensively generated at the edge of the network, transferring the data on such a scale to the distant cloud for processing will slow the system down and lead to unacceptable response time, especially for latency-sensitive applications.

The problems such as low latency, security issues, high price, network problems, and massive data generation from IoT boards make IoT unreliable. The number of IoT actuators and Socket On Chip (Soc) boards have increased in number, which geared for the shift of paradigm with industry 4.0. Industry 4.0 relies on the cloud; there are massive mounts of data generated that is uploaded to the cloud data centre before analysis. Transferring the data at such scale to the distant cloud for analysis increases latency to the network and leads to unacceptable response time, especially for latency-sensitive applications [5].

The failure of IoT to meet the demands of the market has led to the birth of Edge computing. Edge computing is designed to provide flexible computing, network, and storage resources at the edge of the network to support services such as analysis of data offline and also system monitoring[6, 5]. Edge computing, also known as fog computing [4], cloudlets [7], has brought us to the era of computing, considering the amount of data sent to the cloud and transferring the data can be very costly. In this project, the aim is to use edge analytics, analysis near the source of data where is generated [5].

2.2 Objective of the thesis

The main objective is to create a framework for gathering measurement vibration data. In the future this data will be used for, for example, anomaly detection. In this thesis,

the proposed anomalies detection sensors through vibration sensors were BoosterPack from Texas Instruments and BOSCH XDK. The main goal of the project is to have an unmanned engine room, where an operator should not need to inspect the engine room in 30 days but rather the sensors (sound, vibration, oil quality measurements, smell or gas) should replace a human and report anomalies to the system. In general, the aim is to have a predictive maintenance (PdM) system in ship engines.

Background and Related Work

This chapter explains the problem of anomaly detection and analytics through vibration monitoring. Vibration monitoring methodology is not a new methodology. It has been employed in different kinds of industrial analysis for the identification of potential failures and their causes. Most IoT boards fail in terms of computing capability, high latency, filtering, and analytics inability of data gathered from sensors or actuators. The traditional IoT computing devices do not have high-performance capabilities, and also, IoT solutions are only offered to the areas where there is the internet and direct connection to the cloud. These shortcomings hinder deployment to remote areas[8]. Anomaly detection or outlier detection aims at identifying outliers in a dataset, and the outliers are simply data points that are far away from the majority of data points, in the space of data set. The outliers or anomalies usually show a different pattern or behaviour in the dataset than the rest of the data points[9]. The recent advances in telecommunications, the full availability of robust and always-connected smart devices, and the wide adoption of cloud computing services are paving the way towards the concept of edge computing. The common problem in traditional IoT is the availability of effective application layer protocols between service seekers and providers at the devices level, because most IoT devices run on batteries [10]. The dramatic decrease in sensor costs, bandwidth costs, the spread of cellular coverage, and the rise of cloud computing all combine to create favourable conditions to connect devices and mass generation of data efficiently. Vibration monitoring of components in manufacturing involves the collection of vibration data from machine components followed by detailed analysis to detect features that reflect the operational state of the machinery. The analysis leads to the identification of potential failures and their causes and makes it possible to perform efficient preventive maintenance[11].

Effective health diagnostics provide enormous benefits such as improved system safety, reliability, as well as costs for the operation will be reduced, and maintenance of complex engineered systems. Thus, to reduce and possibly eliminate such problems, it is essential to accurately assess the health condition of an operating system in real-time through effective health diagnostics[12].

In the case of real-world factories, from the view of the development cost, it is not

practical to deliberately damage the expensive machine in order to obtain the unusual data.

3.0.1 Sensor Placement

In this project, the sensors are placed on the ship engine, which is in the testing room. The XDK-sensors were placed closer to the coils or springs where there is a higher vibration. The figures 3.1, 3.2 below show where the sensors were placed.

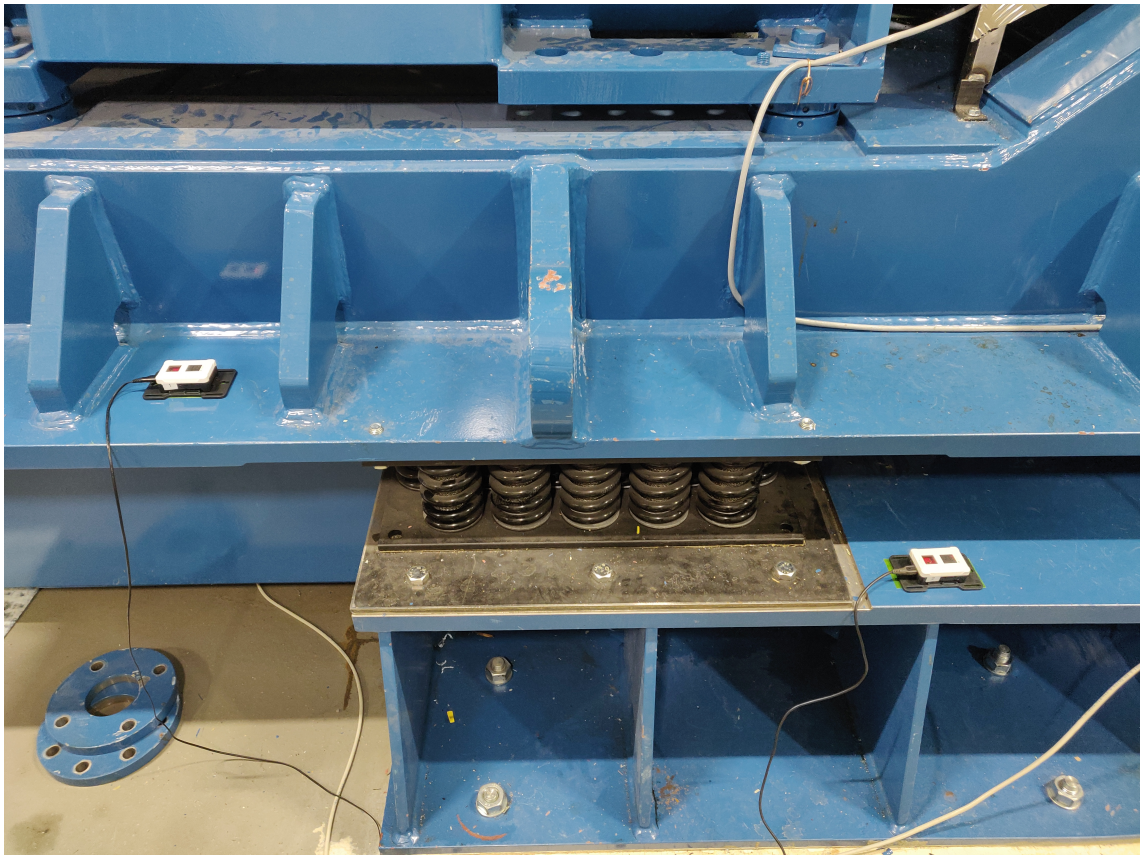


Figure 3.1: XDK placed on the test Engine

3.1 IoT

The internet of Things is about stretching the power of the internet beyond computers and smartphones to a whole range of different things, processes, and environments, it is a system of interrelated computing devices, mechanical and digital machines, objects, animals, or people that are provided with unique identifiers. The connected things collect data. Furthermore,they send information to the cloud or the get-way and from the gateway to the cloud. A thing on the internet of things can be a person, animal, car, or a farm

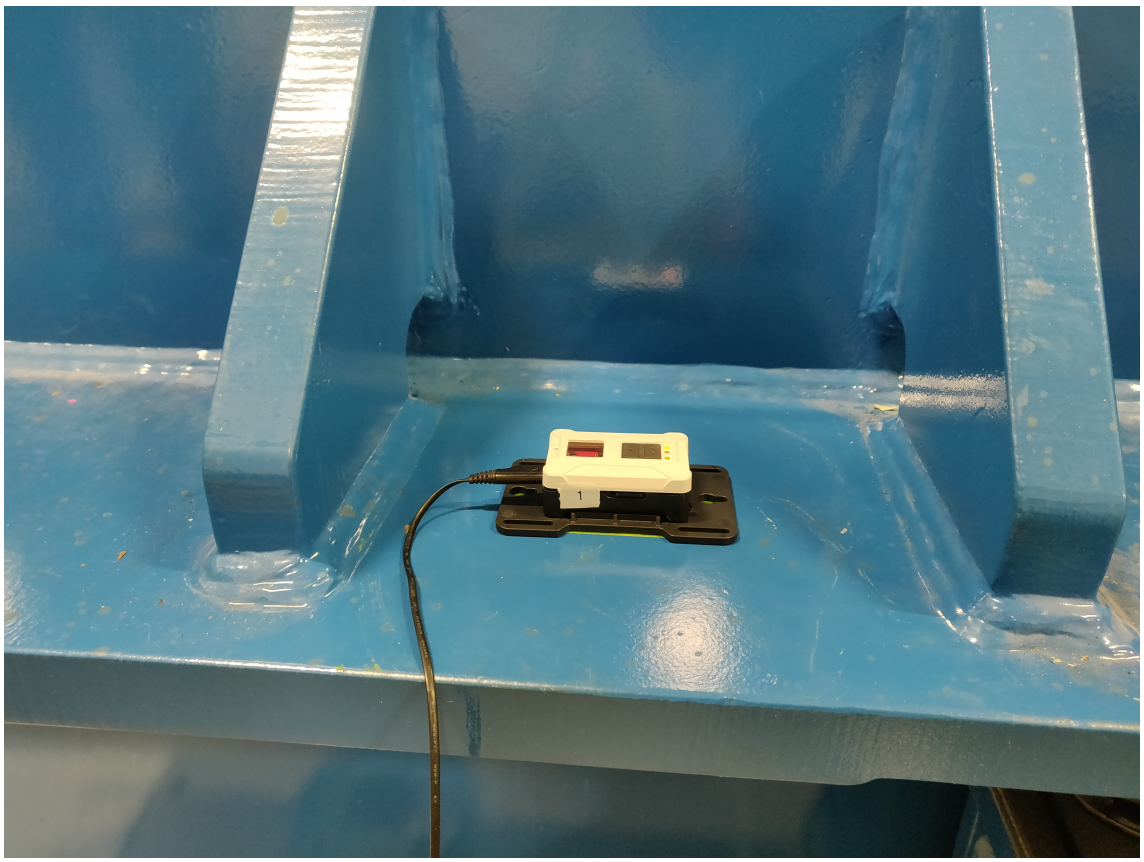


Figure 3.2: XDK placed on the test Engine

with a health monitoring device[13][14].
The internet of Things, which comprises sensors and actuators connected by the networks to computing systems, according to the McKinsey study [15] will have the estimated $11.1 trillion a year in economic value by 2025. In the broader view of the IoT's potentials, researchers [10] [4][16][17] have been developing solutions to fully utilise the power of IoT devices. This ranges from smart cities, health sectors[18][19] [20], Classification of heart sound signals[21] to abnormal heartbeat detection using recurrent neural networks[22].

3.2 Related works in Anomaly Detection

In order to investigate anomalies in Engines, especially diesel engines, researchers have considered and monitored different kinds of signals, such as the vibration signal, acoustic emission signal, current and voltage signals. Nevertheless, this thesis, concentrates Vibration signals generated from the ship engine. Vibrations come as the result of the excitement of different mechanical parts of the engine, and the vibration is not necessarily bad if its amplitude and the associated excitement are within the limits. Engines have bearings, which are the primary source of faults as a result of vibrations; these faults or anomalies can come as the result of material worn out of the bearing. According to researchers, [23] these faults vary from cracks, pits and one of the causes is inadequate lubrication of bearings[23].

3.2.1 Anomaly Detection using Vibration Sensors in Engines

Vibration signal analysis is the most commonly used and practical approach to investigate anomalies in the engine. Researchers Jafarian et al. [24] investigated the engine faults, including the misfire and valve clearance faults, through analysis of vibration data captured by four sensors which are placed in different locations of the automobile engine under different experimental circumstances. Li et al. [25], Hajnayeb et al. [26] researched the gearbox system fault diagnosis, through monitoring vibration of gears and gearbox. These researchers categorised techniques of processing vibration signals in different categories, and they sorted out the raw signal, time-synchronous average signal, residual signal, difference signal, and bandpass mesh signal. They found out after the analysis that the anomalies found changes the amplitude of the gear vibration [27].

3.2.2 Anomaly Detection using Acoustic/Sound Analysis in Engines

Detecting an anomaly or an unusual situation from a given noise is highly useful in an environment where constantly verifying and monitoring a machine is required. According to Oh et al. [16] deep learning algorithms are further developed; current studies have focused on this problem. The researchers proposed model is based on an auto-encoder, and it uses the residual error, which stands for its reconstruction quality to identify the anomaly. They asses the model using Surface-Mounted Device machine sound[16]. Li et al. [28] explored the deep random forest fusion of acoustic and vibratory signals to study the fault diagnosis of the gearbox system. Oh et al. [29] employed the acoustic emission signals to estimate the condition of fan bearings and predict their life before failure. Wang et al. [30] explored the approach of checking the condition of lubrication systems to estimate the extra time of a marine diesel engine.

The researchers Koizumi et al. [17] consider that the set of unusual sounds is the complementary set of normal sounds and simulate unusual sounds by using a rejection sampling algorithm. Through experiments using synthetic data, they found that the proposed method improved the performance measures of the ADS under low FPR conditions. Acoustic emission has several advantages, including rapid inspection of significant components and the capability to determine the location and type of damage [31].

3.2.3 Anomaly Detection in Smart cities

Researchers Bo at al. [18] have explored the use of edge computing with the study of a smart pipeline monitoring system based on fibre optic sensors. Furthermore, sequential learning algorithms to detect events threatening pipeline safety, aiming to secure future communities, these researchers thought it is necessary to integrate intelligence to use edge to perform data representation and feature extraction, to identify anomalous and hazardous events, and to offer optimal responses and controls.

3.2.4 Anomaly Detection in Healthcare

Smart city advancement is driving a massive transformation of healthcare, the largest global industry. Researchers Thaha et al. [32] propose a ubiquitous healthcare framework, and the framework leverages edge computing, deep learning, big data, high-performance computing, and the internet of things.

Presbitero et al. [19] describe methods of anomaly detection in time series data of patients undergoing heart surgery. The methods are metric and model-based indicators and information surprise. These researchers explore the applicability of existing indicators to distinguish anomalies (critical) from average (non-critical) conditions in patients

undergoing cardiac surgery.

Poi and Kok [20] explore the use of fusion sound sensors and accelerometer sensors to increase the fall detection accuracy. They came up with the fall detection algorithm, which implements both accelerometer-based and sound-based detections for the possible occurrence of a valid fall, especially for elderly persons. Nweke et al. [33] also came up with the activity detection algorithm by employing a comprehensive analysis of data fusion. They get data from different sensors of wearable devices.

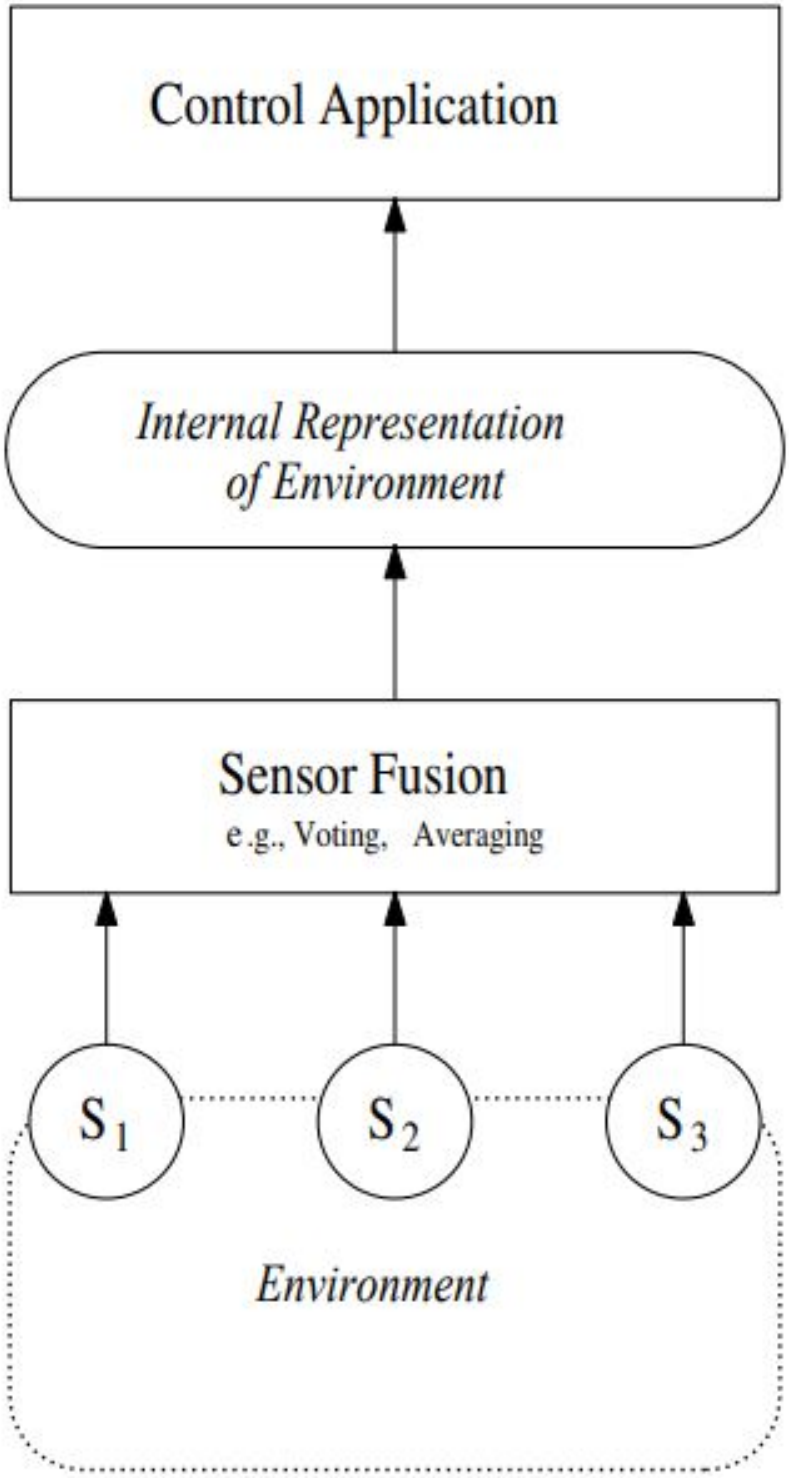
3.2.5 Anomaly Detection in Vehicles

Researchers Long et al. [34] explore the use of mobile edge computing as the solution of the Internet of Vehicles. According to these researchers, the mobile edge computing liberates mobile devices from heavy computation workloads. Ashok et al. [35] and Wang et al. [12] have explored the diagnostic health of vehicles using multi-attribute classification fusion, and these researchers came with the conclusion that it is not possible to rely on one source of health diagnostic of the machine to have the accurate results. They suggest leveraging the strengths of different algorithms to form a robust unified algorithm that could increase the efficacy of health diagnostics.

3.3 Sensor Fusion

In this project, the idea of sensor Fusion came across but was not used at the time of writing thesis. Sensor Fusion is when Multi-sensor data combines data from multiple sensors to perform inferences that can not be done from one single sensor or source alone [36], [37]. Data from different sources and types of sensors are combined using techniques drawn from several disciplines, signal processing, statistical estimation, pattern recognition, artificial intelligence, cognitive psychology, and information theory.[38] Sensor fusion will be deployed in the case of failure to obtain the intended results. Figure 3.3 is an example of sensor fusion.

Elmenreich [38] gives an example that an animal recognises environment by the evaluation of signals from multiple and multifaceted sensors. Nature has a way to integrate information from multiple sources to a reliable and feature-rich recognition. Even in the case of sensor deprivation, systems can compensate for lacking information by reusing data obtained from sensors with overlapping scope. Humans, for example, combine signals from the five body senses sight, sound, smell, taste, and touch, and with that knowledge of the environment interact and update a dynamic model of the world. Based on this information, the individual interacts with the environment and makes decisions about present and future actions[38].



[38].

Figure 3.3: Block diagram of sensor fusion

3.3.1 Types of Sensor Fusion

There are different types of sensor fusions, sensor fusion based on strategies, sensor fusion based on the level of categorisation, sensor fusion based on configuration and sensor fusion based on input or output characteristics. These sensor fusions have different models, and the models include JDL Fusion architecture, waterfall process model, and the LAAS architecture. Sensor fusion has several advantages according to Elmenreich [38]. Multiple sensors increase robustness and reliability, increase confidence when multiple sensors are covering the same domain, and there is a reduction in ambiguity and uncertainty.

3.3.1.1 Based on Strategies

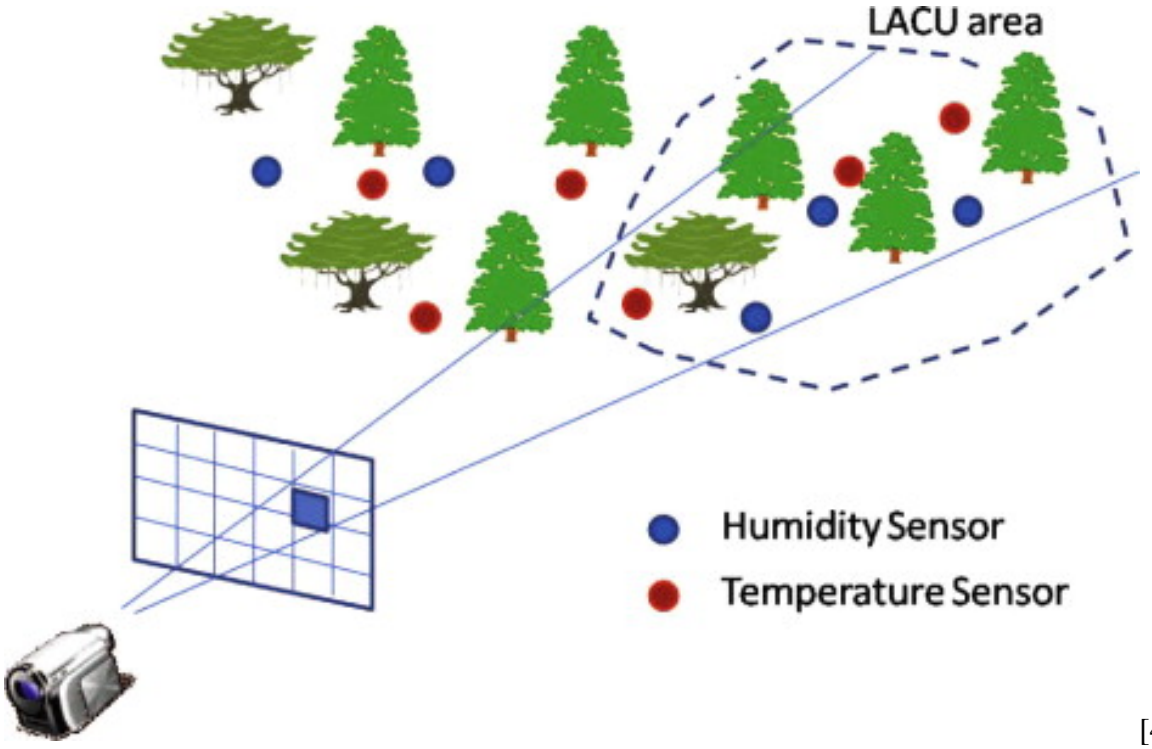
Fusion across several sensors measure the same property, for example, researchers Todin et al. [39] combine block-Kriging, and Kalman filtering as a technique for conditioning radar precipitation estimates to rain-gauge measurements. Figure 3.4 is an example of sensor fusion based on strategies. There is fusion across attributes, whereby several sensors measure different quantities associated with the same experimental situation, for example, researchers Zervas et al. [40] use multi-sensor fusion to measure different fire attributes to improve the reliability of the system. There is also fusion across time, whereby the current measurements are fused with historical information. The current information is not always sufficient to determine the system accurately, and historical information has to be incorporated.

3.3.1.2 Based on level categorisation

Sensor Fusion based on the level of categorisation is categorised in three models. The first is low-level fusion, whereby raw data from different sources are combined to produce new data, which is expected to have more informative than the inputs. The second level is intermediate-level fusion, and this combines various features into a feature map which is then used for segmentation and detection. The third level is high-level fusion, which is the decision fusion, which combines decisions from several experts. Possible methods of decision fusion include voting, fuzzy logic, and statistical methods.

3.3.1.3 Based on Sensor configuration

Sensor Fusion based on sensor configuration is one of the three categories. The first one is complementary configuration. The complimentary configuration is when sensors do not depend on each other but can be put together in order to give a complete image of the phenomena under observation. Complementary sensors can help to resolve the problem of incompleteness[41][38]. The researchers Bennett et al. [42] use multi-spectral



[40].

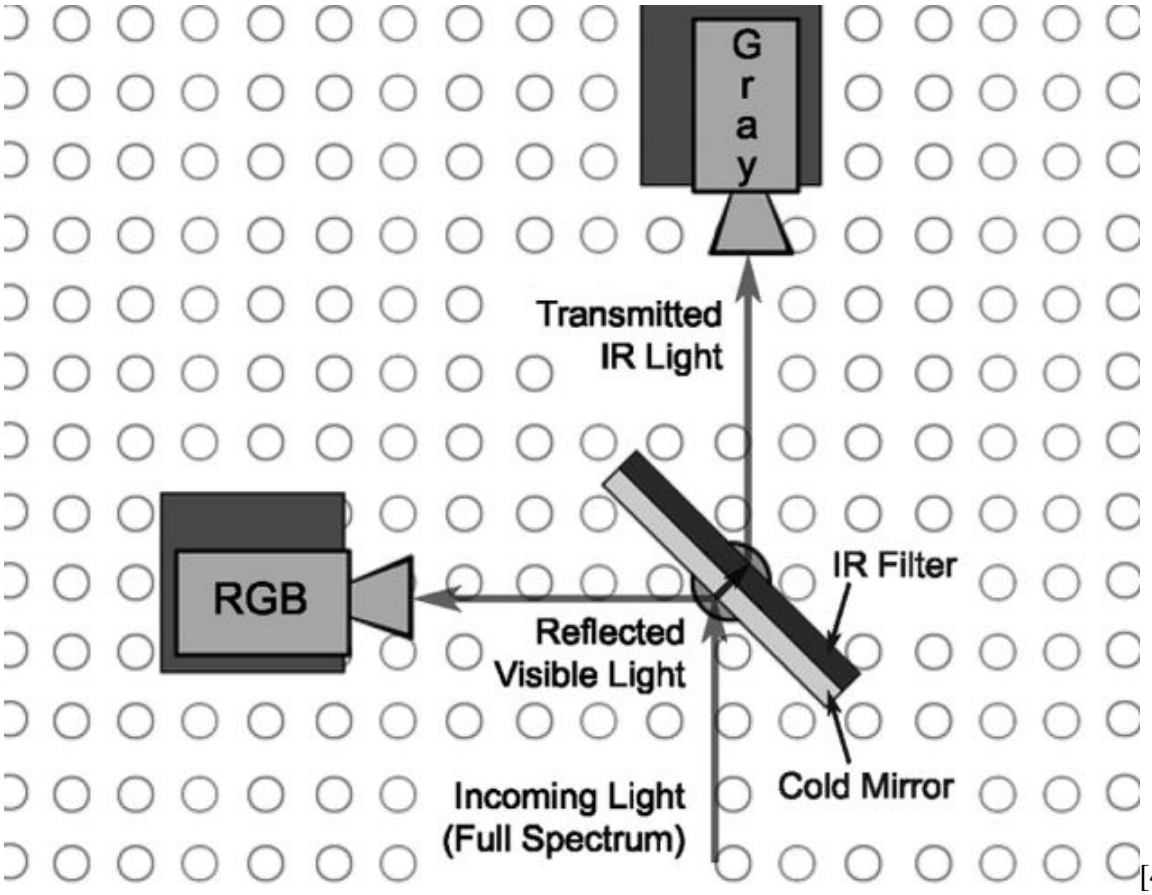
Figure 3.4: Sensor Fusion Based on Strategies

bilateral video fusion to obtain a complete video. Figure 3.5 shows sensor fusion based on configuration, and figure 3.6 shows an example of sensor fusion.

There is competitive configuration when the sensors are configured to deliver an independent measurement of the same property. The aim of competitive fusion is to reduce the effects of uncertain and erroneous measurements[41][38].

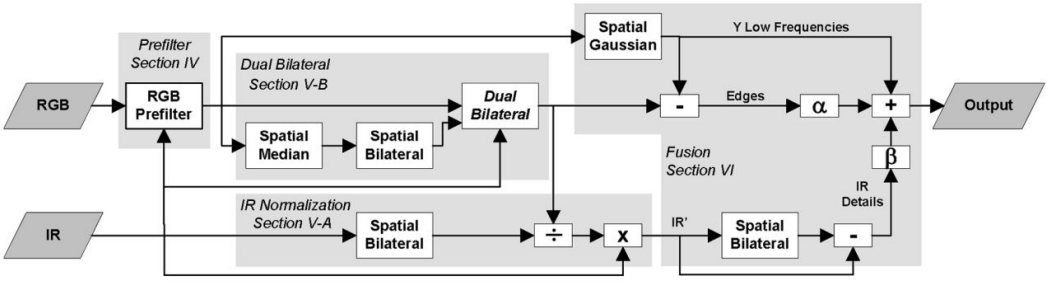
The last configuration according to Elmenreich [38] is a cooperative sensor configuration network which uses the information provided by two or more independent sensors to derive information that would not be available from the single sensors[41]. Researchers Szelisk et al. have used two cameras in stereo vision to combine two input images which view the same scene but are taken from slightly different viewing angles[43] See figure 3.7.

There are three levels of fusion process, the low-level fusion or raw data fusion which combines various sources of raw data to produce new data, the second level is intermediate-level fusion or feature level fusion which combines various features such as edges, corners, lines, textures or positions into a feature map that will be used for segmentation and detection and the last level is decision fusion which combines decisions from several experts including voting, fuzzy-logic and statistical methods[38].



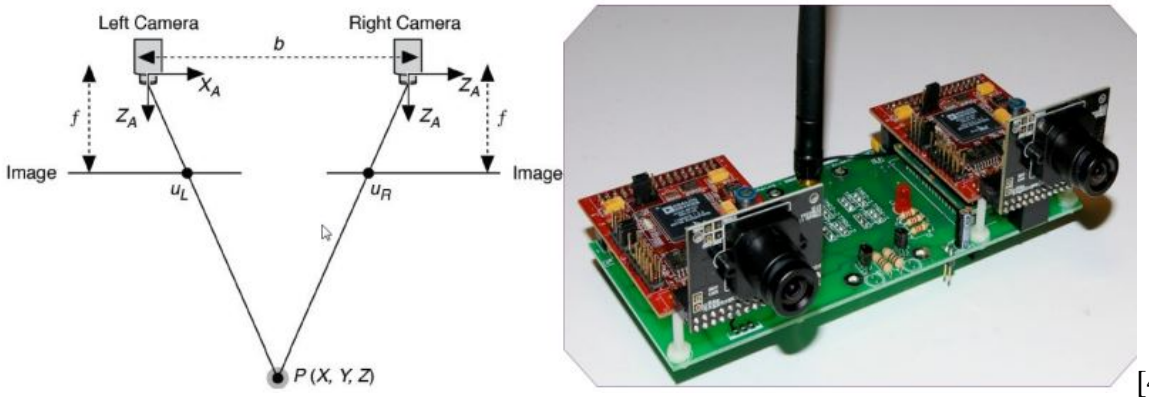
[42].

Figure 3.5: Sensor Fusion Based Configuration



[42].

Figure 3.6: Luminance Fusion Technique



[43].

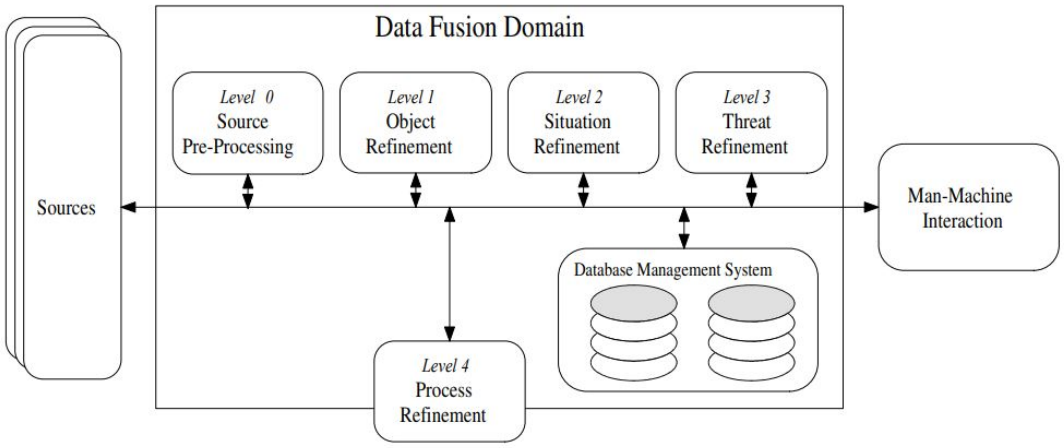
Figure 3.7: Cooperative Fusion

3.3.2 Types of Sensor Fusion Architecture Models

The model originates from the US Joint Directors of Laboratories, JDL. The JDL model comprises five levels of data processing and a database, which are all interconnected through a bus. All levels are not meant to be processed in a strict way and can also be executed concurrently. [38][44][37] There are levels of JDL fusion architecture. See figure 3.8. Sources provide information from a variety of data sources. Source pre-processing, which is level 0, is to reduce the processing load of the fusion processes by pre-screening and to allocate data to appropriate processes. The second level, which is level 1, performs data alignment to reference coordinates and units. The third level, which is level 2, is called situation refinement, which attempts to find a contextual description of the relationship between objects and observed events. The fourth level is called threat refinement and is based on a piece of prior knowledge and predictions. It tries to conclude vulnerabilities and opportunities for operations. The next level process refinement, it is a meta-process that monitor system performance also reallocates sensor and sources to achieve particular mission goals. The database management monitors evaluate, add, and provide information for the fusion process. Lastly, man-machine provides an interface for human input and communication of fusion results to operators and users [38][37].

3.4 Analytics and Anomaly Detection using Edge Computing

This section introduces Vibration sensing for engine condition monitoring. Industry 4.0 has accelerated predictive maintenance in machines, and this helps to minimise operation costs and maximise production. According to Pradeep et al. [45], predictive maintenance



[38].

Figure 3.8: JDL Fusion Model

is one of the essential applications of the Internet of Things. Machine learning aims to predict the possible failure of a machine before the actual event occurs. With the edge computing approach, there is low latency in data transmission. More details will be provided in the next chapter. Anomaly detection is the patterns in data that do not conform to expected normal behaviour[17]. For example, the universal set consists of only the normal and the anomaly, and the anomaly is the complement to the typical set. More intuitively, the universal set is various machine sounds including many types of machines. The usual set is one specific type of various machine sound which is normal, and the anomaly set is all other types of machine sounds which are unknown.

3.4.1 Edge Computing

Edge computing increases the field of information technology beyond the boundary defined by the cloud computing paradigm. Performing computation near the source and destination, edge computing is vowing to address the challenges in many delay-sensitive applications, like real-time human surveillance[46]. Edge computing constitutes a new concept in the computing landscape. It brings the service and utilities of cloud computing closer to the end-user and is characterised by fast processing and quick application response time[47]. With edge computing, there is a larger design space with many parameters rendering a variety of novel approaches feasible. Given the complexity, Edge Computing designs deserve scientific scrutiny for a sound assessment of their feasibility[48]. Zissis explain that edge computing leverages for security reasons which prevent data theft[49]. Dey et al. [50] explains that offloading computationally expensive simultaneous localisation and mapping tasks for mobile robots has attracted significant attention

during the last few years thanks to edge computing. These researchers explore the Cloud Robotics concept, which is extended to include the current emphasis of computing at the source of data. See figure 3.9. Song et al. [51] have explored how access latency has been reduced dramatically because of the decrease in geographical distance in Industry 4.0.

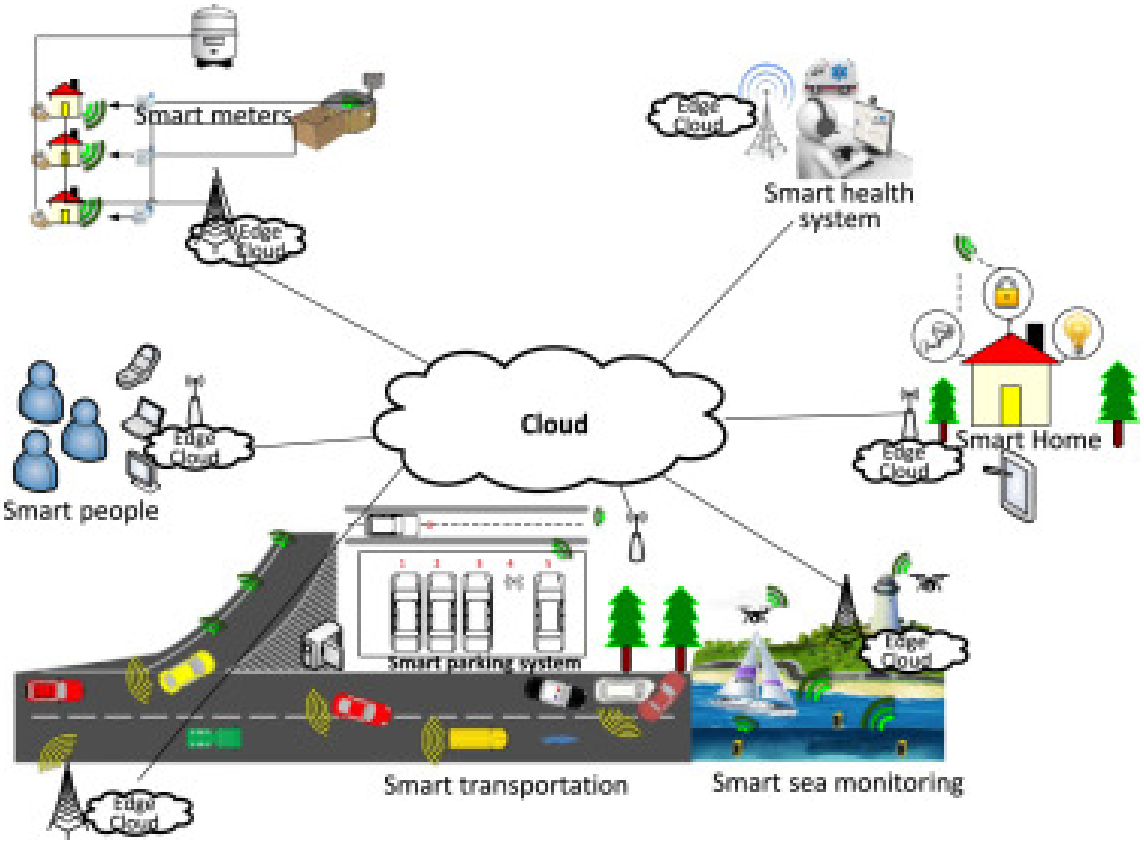


Figure 3.9: Edge Computing Application Example [47].

Truong et al. [52] explain that edge computing models have fostered new types of applications whose software components and dependent services are provisioned across distributed edge and cloud infrastructures. These researchers explore the impact of performance and data quality for mobile edge cloud systems.

3.5 Analytics

Edge analytics refers to implemented analytics that operate either on the device itself or at a nearby device that maintains a hub of units. The analytic processing occurs at multiple locations at the boundary of the network or the edge, versus occurring at a centralised location. In this thesis, we are more concerned with how to analyse the data generated from the sensors.

For the best analytics, we would need a large amount of historical data, but in this project, we lack it. In some cases but not in our case, time is very crucial. When time is recorded for an event, such as a parking spot being vacated, it is essential for analytics that the time is as close to the actual occurrence as possible. In practice, though, the time available for analytics can be the time the event occurred, the time the IoT device sent the data, the time the data were received, or the time the data were added to the edge device. When there are constrained devices there is no network between the devices.This will lead to results in either missing or inconsistent data. The missing data is often not random, and this will lead to misleading analytics results. Also, in some cases, weather factors such as winter storms can lead to power breakdowns affecting devices that can report back data. This may lead to drawing conclusions based on a non-representative sample of data without realising it and can also affect the accuracy, for example, in extremely cold areas[53].

3.5.1 Standard Statistical

Statistical learning is a set of tools for modelling and understanding complex datasets.These tools are classified as supervised or unsupervised[54][55].

3.5.1.1 Linear Regression

In statistics, linear regression aims at providing a linear relationship between the inputs and quantity outputs[9]. Linear regression is a method used to predict a target variable by fitting the best linear relationship between the dependent and independent variables. The best fit is when the total of all the distances between the shape and the actual observations at each point is small. The fit of the shape is best in the sense that no other position would produce fewer errors given the choice of shape. There are two types of linear regression: simple linear regression and multiple linear regression. Simple linear regression uses a single independent variable to predict a dependent variable by fitting the best linear relationship. Multiple linear regression uses more than one independent variable to predict a dependent variable by fitting the best linear relationship[55].

3.5.1.2 Classification

Classification is a data mining procedure where the categories are assigned to a collection of data in order to support more precise predictions and analysis. Also, seldom called a Decision Tree, classification is one of many methods intended to make the analysis of large datasets effective. The primary classification techniques are Logistic Regression and Discriminant Analysis[55]. Logistic regression is a suitable regression analysis to

conduct when the reliant variable is dichotomous. Like all regression analyses, logistic regression is a predictive analysis[55][56].

3.5.1.3 Re-sampling Methods

Re-sampling is the method that consists of bringing repeated samples from the original data samples. It is a non-parametric method of statistical reasoning. The method of re-sampling does not mean the utilisation of the universal distribution tables in order to compute approximate p probability values.

Re-sampling generates a unique sampling order based on real data. It uses innovative methods, rather than analytical methods, to generate the unique sampling distribution. This method produces straight estimates because the method is based on the unbiased samples.

Bootstrapping is another technique that benefits in many situations like the validation of predictive model realisation, ensemble methods, predicting of bias and variance of the model[55][56].

3.5.1.4 Shrinkage

This method fits a model comprising all predictors. However, the estimated coefficients are shrunk towards zero relatives to the least-squares estimates. This reduction or regularisation has the effect of reducing variance. Depending on what type of shrinkage is applied, some of the coefficients may be estimated to be accurately zero. Consequently, this method also implements variable selection. The two best-known techniques for narrowing the coefficient estimates towards zero are ridge regression and lasso. Ridge regression is comparable to least squares except that the coefficients are estimated by minimising a slightly different quantity[55][56].

3.5.1.5 Non-linear Models

In statistics, non-linear regression is a form of regression analysis where observational data is modelled using a non-linear function which is a combination of the model parameters. The method uses one or more independent variables. The data are fitted using a successive approximations method [56][55].

3.5.1.6 Support Vector Machines

Support Vector Machine is a compelling and versatile Machine learning model, capable of performing linear or non-linear classifications, regression. The Support Vector Machines are suitable for classification of complex but small or medium-sized data sets[55][56].

3.5.1.7 Unsupervised learning

Unsupervised learning is applied to determine the data's masked structure pattern in order to learn more from the data. Unsupervised learning aims to detect patterns in the data provided. There are different algorithms which can be used for unsupervised learning, namely clustering, association, and anomaly detection. Clustering is an example of unsupervised learning in which different data sets are clustered into groups of closely related items[9][56][57].

Clustering algorithms are used to identify the inherent groupings in data. The clustering algorithms order the unlabelled data into clusters. Clustering algorithms include K-means clustering, hidden Markov models, self-reorganising maps, Gaussian mixture models and hierarchical clustering[57][9][56]. Principal Component Analysis aids in producing a low dimensional representation of the data set through identifying a set of a linear combination of features which might have maximum variance and are mutually uncorrelated. First, it identifies the hyper-plane that lies closest to the data and then projects the data to it[55][56]. There are other methods, such as the K-means clustering, which partitions data into k distinct clusters based on the distance to the centre of a cluster. Also there is Hierarchical clustering which builds a multilevel hierarchy of clusters through creating a cluster tree[58][56].

3.5.2 Fast Fourier Transform FFT

In the time domain, digital signals describe the signal amplitude against the sampling time instant or the sample number. In this project, we performed digital signal processing to find the signals of interest. The process of transforming the time domain signal samples to the frequency domain components is called a discrete Fourier transform, and this Fast Fourier Transform is the one going to applied in this project. See figure 3.10, which is the example of signal representation, the top plot shows an example of thirty-two signal samples and the lower plot shows the magnitude of the signal spectrum in frequency domain representation[59].

The DFT formula
$X_k = \sum_{n=0}^{N-1} x(n) e^{-i2\pi kn/N}$

The Fourier transform of a signal is generally referred to as the spectrum of the signal, and is used only for periodic signals[60]. According to Mikael et al. [61], the estimation of frequency content in signals is vital in signal processing and systems engineering. According to these researchers, there have been several techniques for estimating harmonic components in signals, and these techniques vary from Hilbert-Huang transformation the

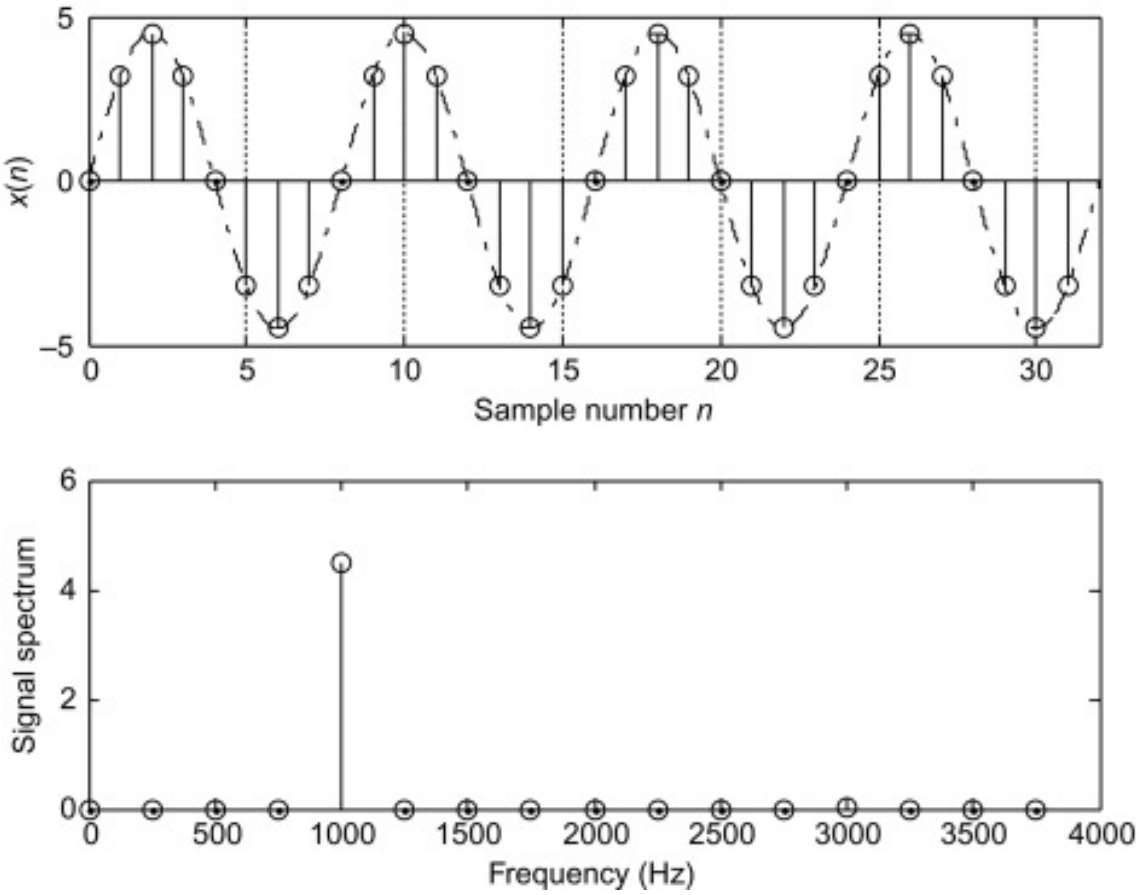


Figure 3.10: Example of the digital signal and its amplitude spectrum [59].

wavelets transform[59] transform and many more. The primary use of FFT in Signals Processing is to find the frequency of interest in a raw data frequency which can have noise and other undesirable components in acquired data. These might be an inherent part of the signal being measured, arise from imperfections in the data acquisition system, or be introduced as an unavoidable byproduct of some Digital Signals Process operation[59].

3.5.3 Artificial Neural Network

The human brain functioning inspires Artificial Neural Networks. They emulate how a human learns. Artificial neural networks are composed of several neurons that are connected. Each neuron is responsible for learning a part of the problem, and afterwards, they communicate that information to produce a final output[9].

Predictive maintenance uses the power of machine learning to detect anomalies. The raw data are fed in the neural network that learns the features automatically. This is where an algorithm is used to find the unique qualities between the signals, whereby we have the input data, and we want the algorithm to produce the output signal from the data. For

example, Artificial Intelligence can extend the life of the engine by combining the exact engine model data, maintenance history, vibration sensors data to detect anomalies, or using images and video of engine condition[62], for example, to train a set of historical vibration data to determine whether specific inputs are out of the ordinary. The system could be trained on a set of data associated with the execution of an operating piece of machinery, and then conclude whether a new vibration reading indicates that the machine is not operating ordinarily [62].

3.6 Data Validation

The time-series data from sensors is a result of multiple processes, from known and unknown processes. Thus, the data creates a problem where one has to identify the processes to be able to minimise the effect of uncertainty on data analyses. Errors in data can occur during collection and also during analysis. For example, the drifting of a timestamp during a process is an error occurring at collection time, another example is a false assumption of states in the analysis phase. These types of errors lead to processed data in giving lousy analysis. There are various methods of data validation which help to show the actual value of data[63]. Among the validation methods are Heuristic rules, Statistical method, Neural network and Sequence of methods[64].

3.6.1 Heuristic rules

In Heuristic rules, there is the minimum and maximum limited window method, Gradient method and State machine. Minimum and Maximum limited window Heuristic rules base their logics in the hope that the designer of the validation methods know about the process, and rules are being applied. They are efficient as they are used as constraints, which gives the method the O-notation, or complexity, of O(n). The limits of the window should be set with expert knowledge of the sensor used, or the environment sensed to ensure that only relevant data is gathered and throw away bad data caused by, for example, sensor fault reading its maximum or minimum value[63].

Gradient Observing the gradient allows us to make sure the data does not change too much within a given time window. This limit is given through expert knowledge to ensure that the limit is reasonable. State machine approach is suggested to validate the system state. The state machine should be given paths that are allowed during regular operation, and maybe complemented with paths that deviate from the standard state path to indicate a fault in the state which has been detected[63].

3.6.2 Statistical method

In Statistical method, there is constant data correlation, comparison of the change between the start and end of idle phases comparison, Change of mean, Z-score, Principal component analysis, and Probability. Correlation may be used both online and offline to validate data because a comparison of two variables is such a trivial task to solve.

Calculating the mean of travels may help to quantify changes in the mean travel-wise. In other words, a mean and standard deviation is calculated using data from a plethora of previous travels to be used as a point of comparison, and the mean of present travel being validated is used together with the two other values to calculate the confidence of the validity of present travel. This methodology may prove useful, as the mean of travels should not have a considerable variation between each other[63].

3.6.3 Hybrid Method

The Hybrid method has the Viterbi method and Digital twin, whereby the Viterbi algorithms estimate the current state by calculating the probabilities of different observations and transitions to different states. The digital twin is a method of having a live digital model running alongside its physical counterpart to validate data acquired by sensors located on, or in the environment of, the physical model. The digital twin is essentially a simulation that gives its own output which represents sensor data of the physical twin, which is expected to be received from its sensors. An algorithm then compares the data from both models and makes a decision, based on different defined criteria, whether the data are reliable or not reliable, and if any action is necessary based on data acquired[63][64].

3.6.4 Neural network method

Neural networks are usually structured based on the problem at hand and then fed a plethora of data differently of labelled data during training and is taught to have certain output for given data. This is generally called supervised learning, whereas unsupervised learning is given blind data during the learning phase, and expecting the neural network to find common variables among the data. These are the most general principles of how neural networks are taught during the learning phase, however, there are a multitude of methods on how neural networks are structured and each may be only applicable to a certain type of problems[63][64].

3.6.5 Sequence of methods

The sequence of methods is applied when several methods of data validation are applied in sequence to provide better results. A sequence of methods may provide better results in detecting different faults or points of interest, as discussed in a paper by Ravichandran et al. [64]. The method presented there uses three methods in sequence to seek erroneous data, a minimum and maximum window for out-of-bounds data, the temporal correlation for constant data, and z-score for outliers and spikes.

3.6.6 Kalman Filtering

Kalman Filter is used to smooth the effects of system and sensor noise in large datasets. It uses a discrete-time algorithm to remove noise from sensor signals in order to produce fused data, for example, to estimate the smoothed values of position, velocity and acceleration of series of point in a trajectory. Kalman filter can be used when one has uncertain information about some dynamic system and wants to make an educated guess about what the system is going to do next. The Kalman filter models can be expressed by two different linear equations, one equation describing the dynamics of the system and one describing the noisy of the system[65].

3.6.7 Principal Component Analysis (PCA)

The method of principal component is primarily a data-analytic technique that obtains linear transformations of a group of correlated variables such that certain optimal conditions are achieved[65][66].

Principal Component Analysis helps in producing a low dimensional representation of the dataset by identifying a set of a linear combination of features which have maximum variance and are mutually uncorrelated. This linear dimensionality technique could help understand possible interaction between the variable in an unsupervised setting[55].

The principal component analysis identifies the direction of the most significant variation in the data. PCA may be useful to estimate the sensor orientation, or if the sensor is loose. In the case of the sensor being loose, the sensor data should have its principal component be different from when it was normal due to the extra movements of the sensor relative to the object it should be fastened onto, thus giving a possible estimate whether the sensor is fastened or loose [63][65].

3.6.8 Sparse methodologies

Modelling sources is critical in signal and image processing. Armed with a proper model, one can handle various tasks such as denoising, restoration, separation, interpolation and extrapolation, compression, sampling, analysis and synthesis, detection, recognition, and more[67][68][69][70].

3.7 Vibration Analysis

In this thesis, we are interested in vibration sensing for engine condition monitoring. Vibration monitoring is based on the principle that all systems produce vibration. When a machine is operating correctly, vibration is small and constant; however, when faults develop and some of the dynamic processes in the machine change, the vibration spectrum also changes [11]. When using vibration to observe machine health, the objective is to correlate observable vibration with typical wear-out mechanisms, such as bearings, gears, chains, belts, brushes, shafts, coils, and valves. In any machine, the mentioned mechanisms require regular maintenance[71].

The analysis leads to the identification of potential failures and their causes and makes it possible to perform efficient preventive maintenance. Early Detection is vital because it can decrease the probability of catastrophic failures and reduce maintenance costs [72].

The progress of any monitoring program depends on the accuracy of the measurements that affect directly the ability to detect and identify faults. Given that the instrumentation is calibrated correctly, measurements are accurate when the sensor mounting does not limit the frequency and dynamic ranges of the sensor, when the sensor mass does not change the dynamic characteristic of the object, and when measurements are always collected at the same locations [23]. The analysis leads to the identification of potential failures and their causes and makes it possible to perform efficient preventive maintenance.[11].

Accelerometers are a popular transducer for vibration analysis because of their accuracy, lightweight, excellent temperature resistance, and comprehensive frequency response[73, 74]. Delgado et al. [74] accelerometers provide an excellent mechanism for collecting vibration data; they are also highly sensitive to noise. These researchers studied that most of the electrical faults are results of worn-out bearings, and this is because of material fatigue. The faults caused by bearing worn-out caused different vibration in the machine[11]. Delgado et al. [74] studied the use of Neural networks as the way fault identification through analysis of vibration data from the defected machine. They use feed-forward neural networks for classifying healthy and defected bearings[75].

3.8 Summary

This chapter explained the problem of vibration monitoring, giving a background to where the experiment will be done. This chapter also gave a background to IoT and where it stands in this project. It also explained what is anomaly detection and highlighted examples of anomaly detection. In this chapter, also an explanation of sensor fusion and how it can be integrated into the project was provided. Furthermore data validation methods were discussed since we are going to deal with data, and the data analysis methods. In the next chapter, the components which will be used in this project will be introduced.

Hardware Components

This chapter explains the implementation details. Section one explains the physical components which will be used in the experiment to correct data from the engine and how these devices are placed on the engine. The engine is Wärtsilä 4L20, a four-cylinder 4-stroke diesel engine. It produces a maximum of 800kW of electrical power at 1000 rpm. Section two explains the edge device where data is stored, processed/analysed. Section three explains the Software layer, which is the programming language used implementing the algorithms C, C++ and Python are selected as the primary programming languages. The XDK workbench, which is based on Eclipse IDE, is used for implementing the overall program of the project. We chose Bosch XDK over Texas Instrument BoosterPack because XDK is a ready-made industrial product and it can sustain up to 60°C environment.

4.1 Hardware Stack

This section describes the physical devices needed in the project. In this study, the devices which are used are Texas Instrument BoosterPack, Bosch XDK110, Adlink computing device and Diesel Engines. The aim of the thesis was to find finds anomalies by depicting engine failures based on the deviations from healthy data[76].

4.1.1 Accelerometer

An accelerometer is an electro-mechanical device that measures acceleration forces. The accelerometer has many different parts and works in many ways. These parts which make the acceloremter sensors are the piezoelectric effect as well as the capacitance. The piezoelectric effect is the common form of accelerometer and uses microscopic crystal structures that become stressed due to accelerative forces. Also the crystals create a voltage from the stress. The accelerometer reads the voltage to determine velocity as well as orientation[77][78].

The capacitance accelerometer senses variations in capacitance between micro-structures located next to the device. If an accelerative force happens to move one of these struc-

tures, the capacitance will always change, and also the accelerometer will translate that capacitance to voltage for interpretation[77] [78].

MEMS accelerometer sensors are the ones used in this project for prototyping. Traditionally the vibration sensing instruments are of piezoelectric technology, but with fast recovery after great shock, low power, small in size and low weight make the MEMS sensors the natural choice. MEMS vibration sensors are stable and can withstand various temperatures.

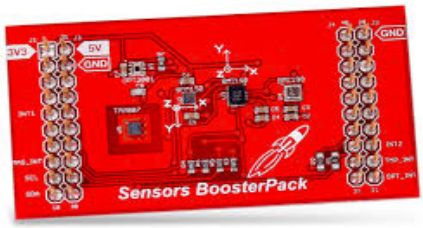
Accelerometers are of multiple axes. Smartphones typically use three-axis models, while cars use only a two-axis model to determine the moment of effect. The sensitivity of these devices is quite powerful as they are intended to measure even every shift in acceleration. The more sensitive the accelerometer, the more quickly it measures acceleration[77][78]. The acceleration comes from three axis X-Y-Z.

4.1.2 Texas Instrument BoosterPack, and Bosch XDK110

The BMI160 is inertial measurement consisting of state of the art three axes, low-g accelerometer and a low power three-axis gyroscope. Gyro has been designed for the use of low power devices. High precision six-axis and nine-axis applications are found in mobile phones, tablets, wearable devices, remote controls, game controllers, head-mounted devices and in many toys.

4.1.3 Texas Instrument BoosterPack

In this project, we also had Texas Instrument BoosterPack. The Sensors BoosterPack is an easy-to-use plug-in module for adding digital sensors to a development board. It can be used to develop sensor applications using the on-board gyroscope, accelerometer, magnetometer, pressure, ambient temperature, humidity, ambient light, and infrared temperature sensors. [79]See figure 4.1.



[79]

Figure 4.1: BOOSTXL-SENSORS BoosterPackK

The BMI160 is a well-integrated, low power inertial measurement unit (IMU) that

provides precise acceleration and angular rate (gyroscopic) measurement. [80] The sensor data from accelerometer and gyroscope are synchronised on the hardware level, which means that they run on the same sampling rate. [79] The BMI160 supports various levels of data synchronisation, for example, the internal synchronisation accelerometer of gyroscope and accelerometer and external sensor data. For the data processing in the gyroscope sensor, it is supposed to operate in normal power mode[79].

Typical applications of BMI160 are Augmented reality, 3D scanning/indoor mapping, Advanced gesture recognition, Immersive gaming, 9-axis motion detection, Free-fall detection and Warranty logging [79].

I2C protocol or Inter-Integrated Circuit is abbreviated as I2C. Raspberry Pi3 which was used earlier in the project has I2C bus uses to communicate with the CPU, and the booster pack sensors were connected to the I2C. I2C is a multi-master, multi-slave, the serial computer invented in order to simplify the board schematics. The electrical lines of I2C bus are SCL-Serial Clock which is the bus for the clock signal, SDA-Serial Data which is the bus for data signal and GDN is common ground. Each I2C device has a well-defined 7 bits address that the master must use in order to communicate with a device[81].

4.1.4 BMI160 Data Processing Accelerometer

The BoosterPack has MEMS accelerometer sensor. This accelerometer can be operated in both normal power mode and low power mode. In normal power mode, the output data rate can be configured in one of eight different valid output data rate going from 12 Hz up to 1600 Hz. The lower power mode the output data rate can be configured from 0.78 Hz to 1600 Hz[79].

4.1.5 BMI160 Data Processing Gyroscope

The BoosterPack also has a gyro sensor. The gyroscope data is processed in normal power mode only. The output data rate can be set in different eight valid output data configuration rate going from 25 Hz up to 3200 Hz[79].

4.1.6 XDK

The Bosch Cross Domain Development Kit is a programmable sensor device for building IoT applications. The XDK110 is a wireless sensor device. The XDK110 gives the opportunity for users to explore more advanced programming on the device. The operating temperature ranges from -20 °C to 60 °C[82].

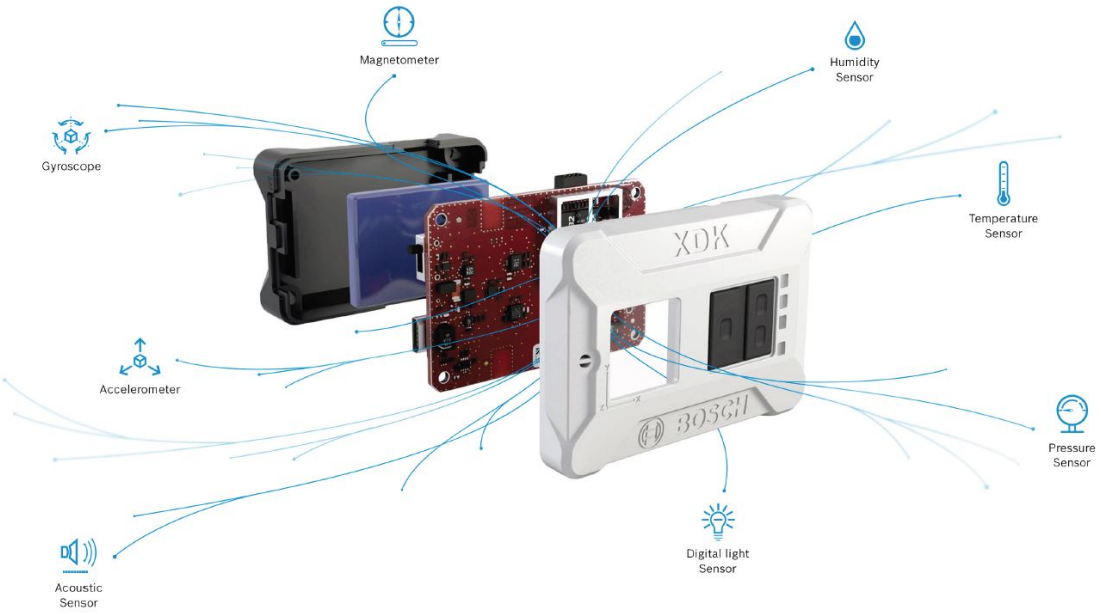
XDK contains a wide range of sensors and means of connectivity and is extensible using its extension bus. Due to its versatility, it also serves as a reference platform for

Eclipse Mita. LWM2M communication protocol, extensive libraries and modular sources code help the developer to utilise the system. The user interface of the XDK has a power switch, and green system LED to display the state of charging, three programmable LEDs (red, orange, yellow), two programmable push-buttons, Interface for J-Link Debug-probe, Interface for extension board. The sensors sampling rate is 2000 Hz for Accelerometer BMA280, and Gyroscope BMG160 for the Magnetometer BMM150 is 300 Hz.

The XDK can be programmed using C language in its programming environment(IDE) which is eclipse based IDE, and it can be also be programmed using Eclipse Mita which is a programming language that is focused on making IoT boards easier to program, especially for developers without an embedded development background.

4.1.7 XDK110 Sensors

The built-in sensors of xdk include accelerometer, acoustic sensor, digital light sensor, gyroscope, humidity sensor, magnetometer sensor, pressure sensor and temperature sensor, inertial sensors which include Accelerometer BMA280, Gyroscope BMG160, Magnetometer BMM150 and Inertial measurement unit BMI160. There are also environment sensors which are in the xdk, Microphone for noise detection AKU340 (Akustica), Ambient light MAX44009, Combined humidity, temperature, and air pressure sensor BME280 [82]. See figure 4.2.



[82]

Figure 4.2: Sensors on XDK

4.1.7.1 Accelerometer BME280

The BMA280 is an advanced, ultra-small, triaxial, low-g acceleration sensor with digital interfaces, aiming for low-power consumer electronics applications. Featuring 14-bit digital resolution, the BMA280 allows very low-noise measurement of accelerations in three perpendicular axes and thus senses tilt, motion, shock and vibration in cellular phones, handhelds, computer peripherals, man-machine interfaces, virtual reality features and game controllers. There are also other sensors, such as Environmental sensor, Geomagnetic sensor, Ambient Magnetic Light Sensor, but in this project, we are interested only in vibration sensors [82].

4.1.7.2 Gyroscope BMG160

The BMG160 is an angular rate sensor with a measurement range up to 2000 degrees per second and a digital resolution of 16 bit for consumer electronics applications. The BMG160 allows low-noise measurement of angular rates in three perpendicular axes and is designed for use in different devices [82].

4.1.7.3 Inertial Measurement Unit BMI160

The BMI160 is a small, low-power, low-noise 16-bit inertial measurement unit designed for use in mobile applications like augmented reality or indoor navigation which require highly accurate real-time sensor data. In full operation mode, with both the accelerometer and the gyroscope enabled, the current consumption is typically 950 Micro Amps, enabling always-on applications in battery-driven devices [82].

4.2 Communication Protocols

XDK is equipped with different types of radio communication, and this includes state of the art low-power WLAN transceiver and Low Energy Bluetooth[82]. The Wi-Fi API provides several interfaces to manage the Wi-Fi functionality on the XDK. XDK applications can implement this API to communicate over the Simple link stack with surrounding Wi-Fi networks. All network information such as IP-address, connection state, gateway or subnet mask is available. There are API features for scanning and initiating several functionalities. There is HTTPS connection protocol also which used in this project for connection between the XDK and the edge device. HTTP tells how messages between two network endpoints are supposed to work so as to obtain an efficient data transfer. In every interaction, one network endpoint acts as the client and the other as the server.

The server listens for incoming requests and serves them as they arrive. For all the requests, the server replies. The XDK also has MQTT messaging protocol which is the communication of machine to machine[82].

4.2.1 UDP

With UDP, computer applications can send messages, in this case, referred to as datagrams, to other hosts on an Internet Protocol network. Prior communications are not necessary in order to set up communication channels or data paths. UDP uses a simple connectionless communication model with a minimum of protocol mechanisms. UDP provides checksums for data integrity, and port numbers for addressing different functions at the source and destination of the datagram. UDP has no handshaking dialogues, so it exposes the user's program to any unreliability of the underlying network. UDPprovides the minimal functionality needed to use the IP's raw datagram delivery services. A UDP message contains a pair of source and destination port fields, a length field and an optional checksum that verifies the entire UDP [83]. UDP is suitable for purposes where error checking and correction are not necessary for the application; UDP avoids the overhead of such processing in the protocol stack. Time-sensitive applications often use UDP because dropping packets is preferable to waiting for packets delayed due to re-transmission, which may not be an option in a real-time system [84]. UDP is scanning is susceptible to failure, and it easy for results to be misinterpreted[83].

4.2.2 MQTT

MQTT is one of the most commonly used protocols in IoT projects. It stands for Message Queuing Telemetry Transport. Besides, it is designed as a lightweight messaging protocol that uses publish/subscribe operations to exchange data between clients and the server [85].

4.2.3 Lo-RA

Lo-RA or Long Range is a low-power wide-area network technology. It is based on spread spectrum and modulation techniques derived from chip spread spectrum technology.

In this study, we are going to use XDK an boosterpack, the operating measurements for the Accelerometer sensor range from ±2 g to ±16 g, Bowers [72] explains that vibration levels recorded at each sensor for each machine differ depending on the fault. The vibration levels may also differ because of sensor positioning and overall neigh-boring vibration. These divergences among sensor readings make it difficult to produce the required vibration signature for a particular fault and type of machine. [72] To address this

problem in this project, we decided to use two XDKs to have data from two different sensor readings. See figures 4.3, and 4.4.

Figure 4.3: The XDK110

4.2.4 Edge Device

In the VEBIC lab, the installed edge device in which data is accumulated and analysed is an Industrial computing unit called Adlink MXE-5401/M16G. We installed Centos 7 as the operating system. The processor of this computing device is Intel Core i7-4700EQ 2.4. It can be wall-mounted. The operating temperature is -20°C to 60°C. See figures 4.5 and 4.6.

4.3 Summary

In this chapter, we explained the hardware components used in this project. The elements include the development boards, sensors used such as MEMS, and also the communication protocols WLAN, UDP, MQTT. Also edge device is explained. The following chapter will explain the experimental design and implementation of the project.

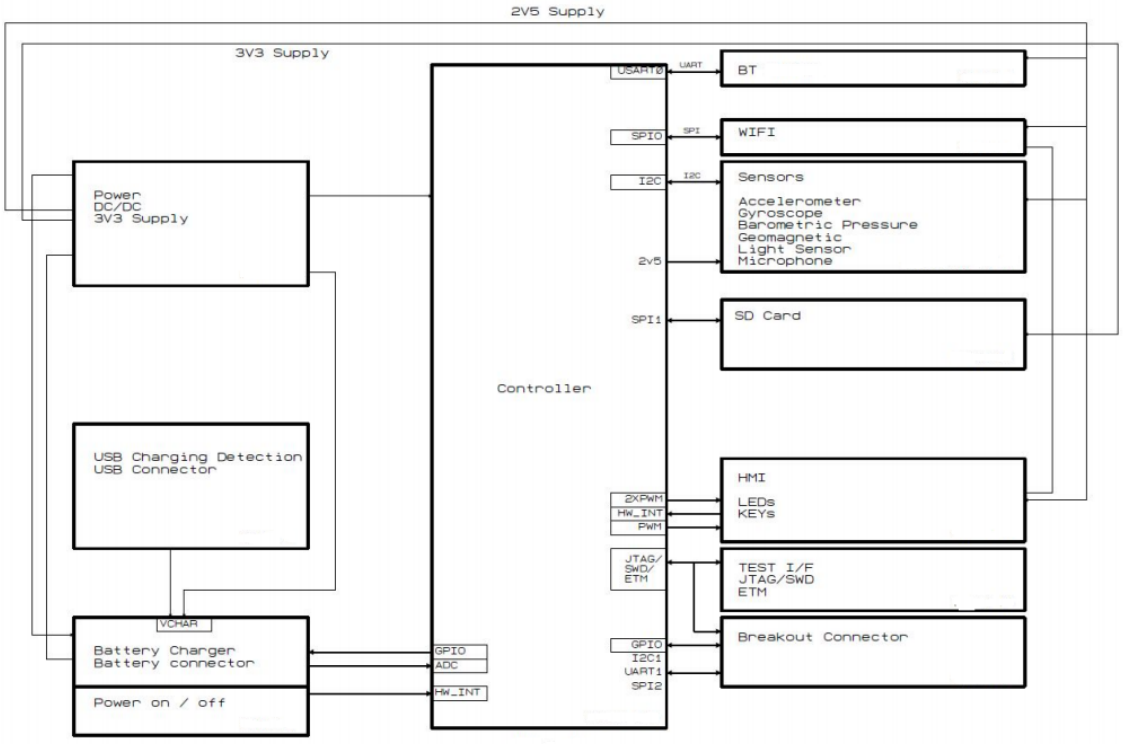
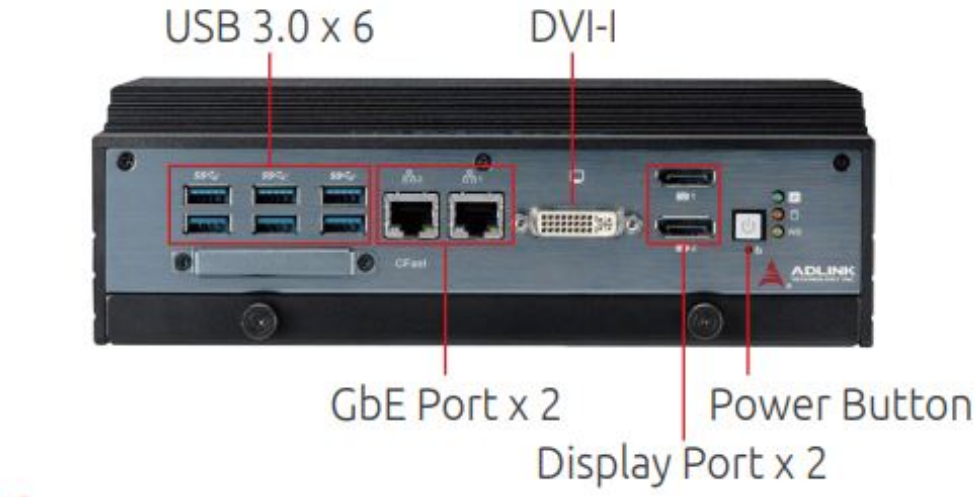
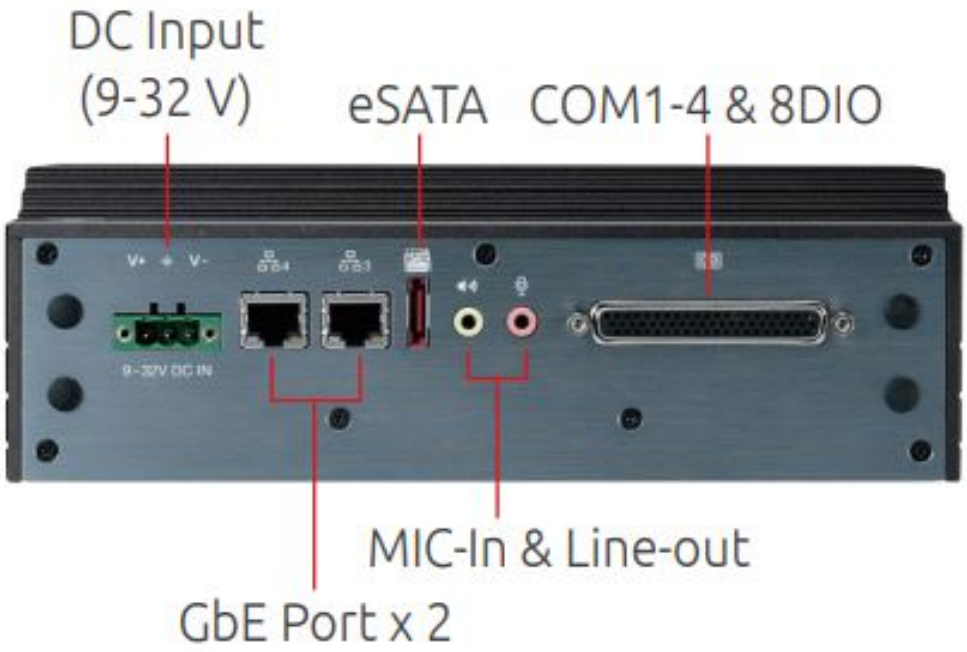


Figure 4.4: XDK110 Block Diagram



[86]

Figure 4.5: Adlink Edge device Front side



[86]

Figure 4.6: Adlink Edge device Back Side

Experimental Design and Implementation

This chapter explains how the experiment was conducted and the results. The section explains where the test was conducted, and it gives a brief explanation of VEBIC lab as the experiment area. Part two explains the sensor placement on the engine. Part three explains data storage from the sensors data analysis which is explained in the next chapter.

5.1 Engine Lab of VEBIC

Vaasa Energy Business Innovation Centre or VEBIC is a research and innovation platform which is hosted by the University of Vaasa. It brings together know-how from the research and business alliances responding to the global necessities of efficient energy production, energy business, and sustainable societal development. Energy and sustainable development are one of the new core focus areas of the University of Vaasa. As an extensive research infrastructure, VEBIC has a central role in realising the new approach.

The VEBIC platform includes laboratories and a program for energy and sustainable development research projects. There are two laboratories, an internal combustion engine laboratory and a separate but interconnected laboratory for fuel development[87]. The engine is Wärtsilä 4L20, a four-cylinder 4-stroke diesel engine. It produces a maximum of 800 kW of electrical power 1000 RPM; It is in VEBIC lab[88].

With its VEBIC research infrastructure, the University of Vaasa inquires to thicken its position among the international leaders in the realm of energy research. The VEBIC serves international and domestic research and education needs of the energy industry and academic institutions. It will also offer a unique, extensive research infrastructure for businesses and other universities.

5.2 Sensor Placement on Wärtsilä 4L20 engine in VEBIC

The lab is the VEBIC building where the testing engine is. Two XDKs were placed on the engine to collect the data, and the data were collected and stored in the adlink edge device stored influxdb for analysis.

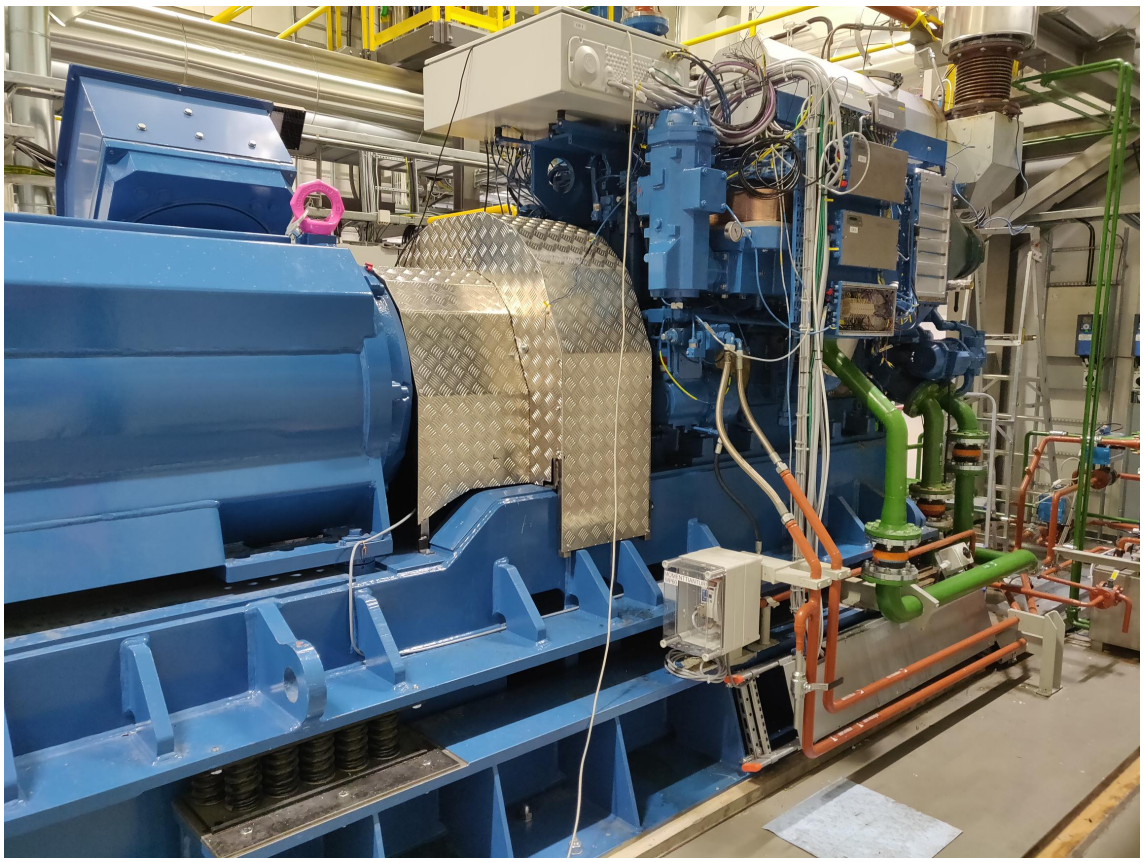
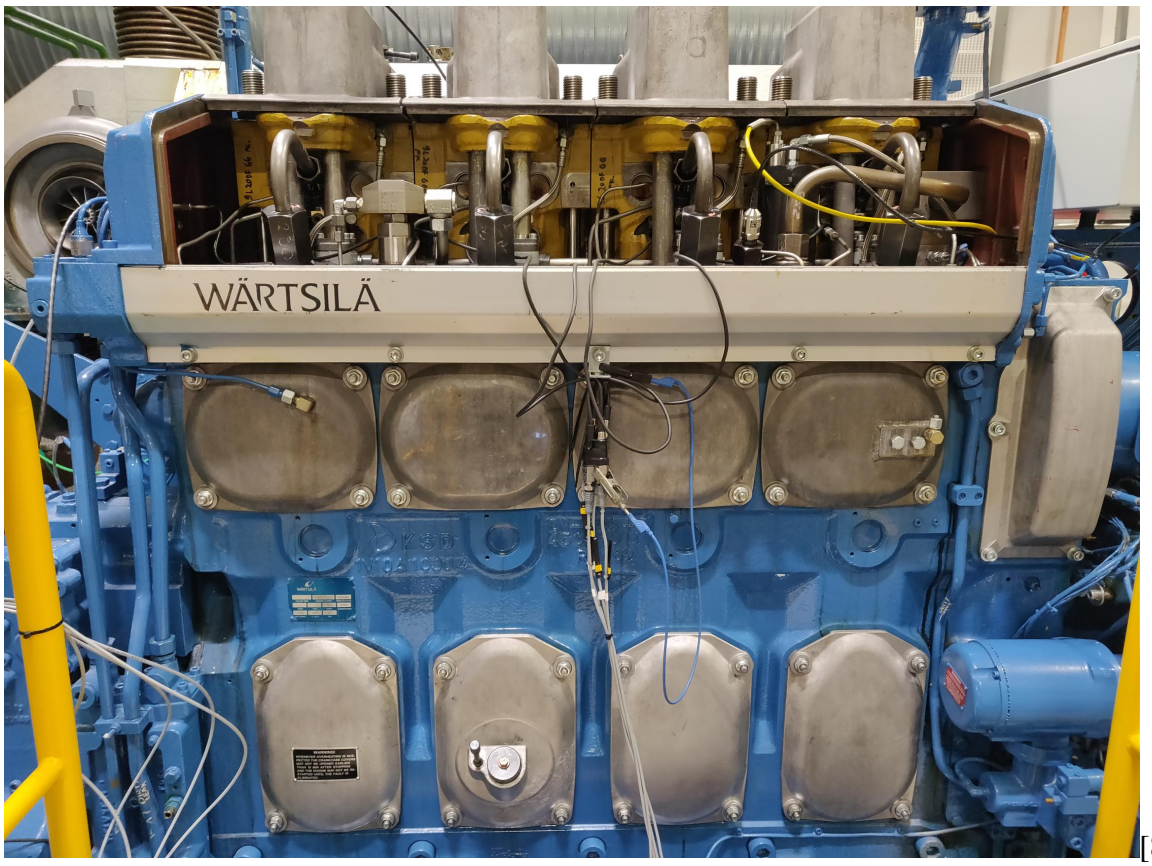


Figure 5.1: Wartsilä 4L20 Engine in VEBIC Lab



[88].

Figure 5.2: Wartsilä 4L20 Engine in VEBIC Lab

[88].

Figure 5.3: Wartsilä 4L20 Engine in VEBIC Lab

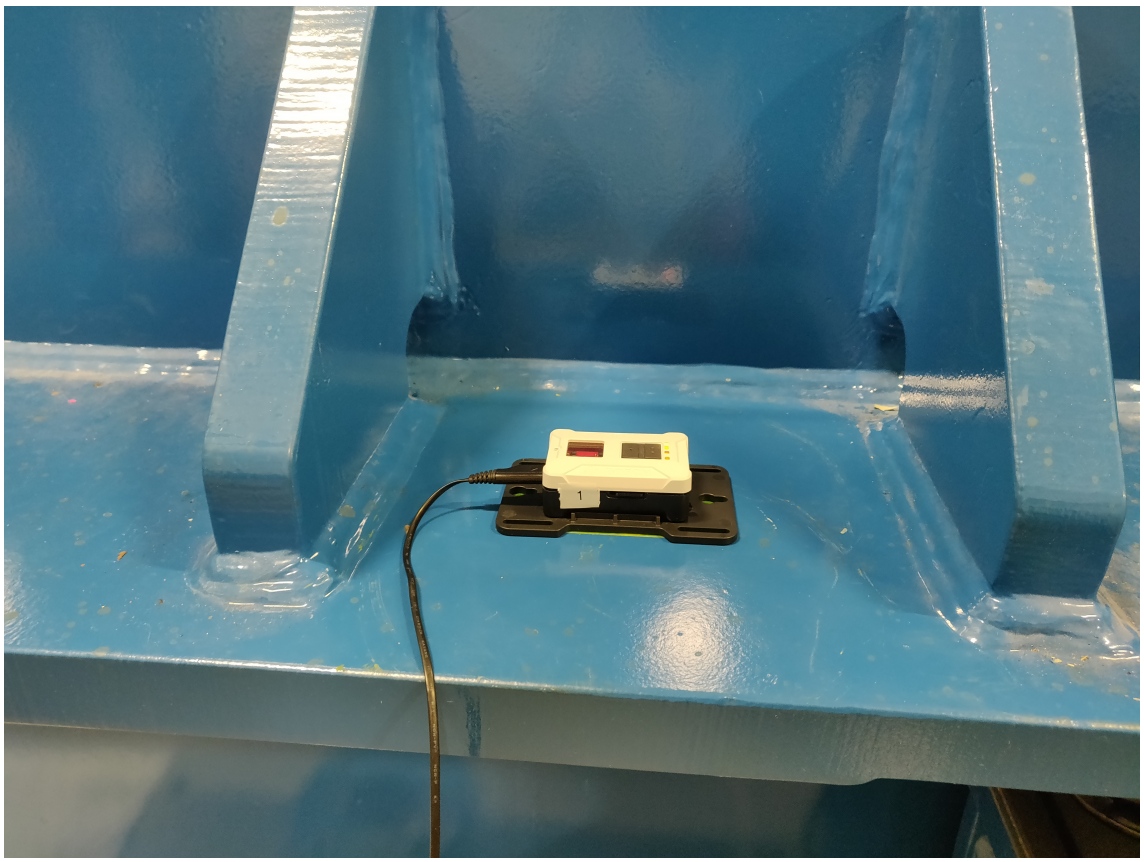


Figure 5.4: This figure presents Test Engine

5.2.1 Data Storage in VEBIC

The regular engine data storage is done by LabTool, which has one dedicated computer in the control room. If we think about the EDGE-project only, the vibration data are stored in InfluxDB 2.0, which is installed on Adlink. The information from VEBIC is that they have plans on gathering engine data directly on Adlink MXE5401/M16G, but those data will also be stored in the InfluxDB or similar system in the end.

5.3 Experiment with BoosterPack Wasaline

There was an experiment on the ship, which is Wasaline. The setup for the experiment is Raspberry Pi, which has a program of data acquisition from booster pack sensors. See figure 5.6 which represents the setup in Wasaline and 5.5 schematic diagram of Raspberry-pi and booster pack.

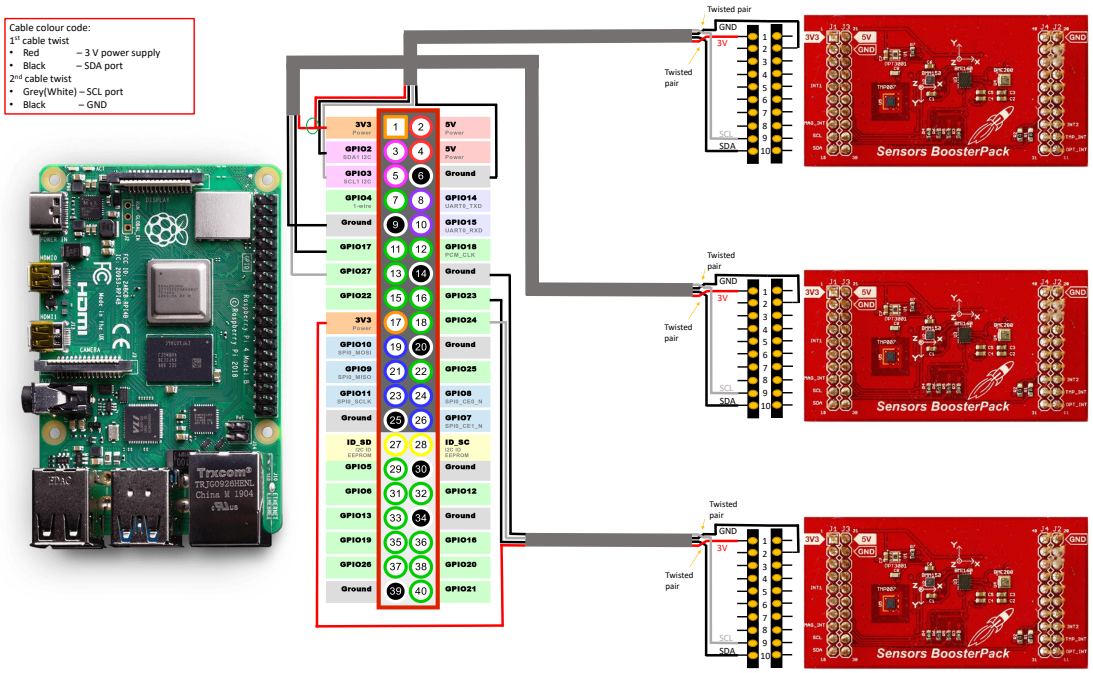


Figure 5.5: The Wasaline Schematic Setup

5.3.1 Raspberry Pi Setup

The YOCTO project was used to set up the Raspberry Pi. The idea of using YOCTO came after having problems with the Pi shutting down during the testing time. We came to the conclusion to use customised image. The solution was to use the YOCTO project, but the

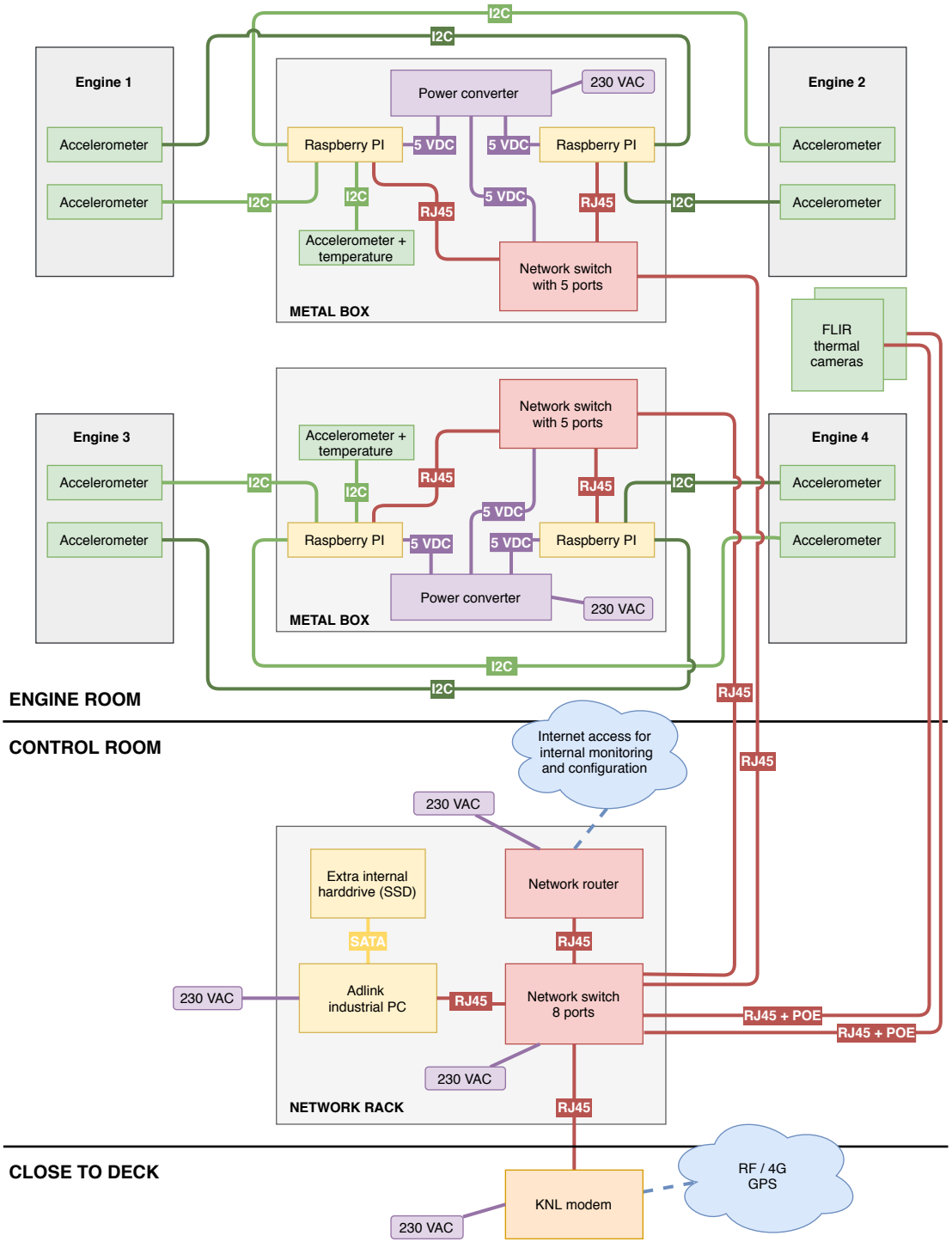


Figure 5.6: The Wasaline Setup

Yocto project image was abandoned for a while because of the Vmware technical problems. The Yocto Project is an open-source collaboration project that provides templates, tools, and methods to help us create custom Linux-based systems for embedded products regardless of the hardware architecture. One of the resources is OpenEmbedded-Core, which is the main core of the system component. There are also other components, such as Poky. There are different Layers which can be customised according to the need of the project.

5.3.2 Testing the Frequency

The FM radio was used on the Raspberry Pi to test the signal disturbances. The reason for that was to create an environment similar to that of the ship. In the ship there are electromagnetic frequencies, so we decided to do the setup which makes electromagnetic disturbances.

5.3.3 Installation of InfluxDB and Chronograf on Raspberry Pi

Influx DB was installed on the Raspberry Pi for data storage. The analysis will be spectral analysis through chronograf which was also installed. The spectral analysis offers a convenient way to characterise the vibration profile, which is the relationship between vibration magnitude and frequency.

5.3.4 Storage of Data

The data storage was on an external hard drive. The external drive was Samsung T5 SSD 500GB, which is using the ex-FAT file system. This file system is used mostly in windows, and to make it work in our environment, which is Linux, we had to install the ex-FAT driver for the system to work, and mounting is not required.

5.4 Summary

In this chapter, we explain how the experimentation and implementation were done. The chapter explains the Engine Lab of VEBIC where the case study using XDK sensors was done. Also, we explain the engine specification and where the sensors are placed on the engine. Also, in this chapter, we describe the implementation and experiment, which was to be done on the Wasaline ship. This explains the installation of the InfluxDB and Chronograf on Raspberry Pi and how the data are stored. The following chapter explains about data collection and Analysis.

Data collection and Analysis

This chapter explains the data type, storage, and data analysis. The section explains the software stack and the data. There are subsections which describe docker container and docker-compose. There is also a section describing the influxdb.

6.1 Acceleration Data

The acceleration data from MEMS sensors BOSCH XDK's and Texas Instrument BoosterPack were stored in the respective databases for analysis in real time. The database used was influxdb, and the tools or software for analysis were grafana. The Wasaline case, the database, and the analysis tools/software were installed in Raspberry Pi, and the VEBIC case or Wartsila was installed in the adlink.

6.2 Software Stack

The XDK comes with an SDK that offers a wide variety of libraries and tools to application developers. The part of the SDK that covers networking protocols is called the Serval stack. It contains modules for a range of application layer protocols that are used on embedded systems. The XDK platform offers many useful high-level APIs. When dealing with essential topics like task scheduling or inter-task communication, though, we have to resort to the functions offered by the underlying operating system. The operating system that powers the XDK is called FreeRTOS. It is a lightweight, open-source real-time operating system, built specifically for embedded systems [82]. The time-series data are accumulated in Adlink [86] from the XDK, the communication between the Adlink and XDK is done through UDP. The reason of using UDP is to minimise latency and also because it enables processing communication [89]. InfluxDB[90] is installed on Adlink. Adlink is a high-performance time-series database. In order manipulate time-series data, we used TICK stack, which is a platform of open-source tools built for collection, storage, graphing, and alerting on time-series data [90].

6.2.1 Docker

Docker is a set of platform-as-a-service products that use OS-level virtualisation to deliver software in packages which are known as containers. Containers are secluded from one another and bundle their software, libraries, and configuration file. A Docker container image is a lightweight, standalone, executable package of software that includes everything needed to run an application: code, run time, system tools, system libraries, and settings. Containers isolate software from its environment and ensure that it works uniformly despite differences, for instance, between development and staging[91]. Docker containers can be portable anywhere. They are lightweight containers share the machine operating system kernel and therefore do not require an OS per application. Docker containers are safer in containers, and Docker provides the most robust default isolation capabilities in the industry[91].

6.2.2 Docker-compose

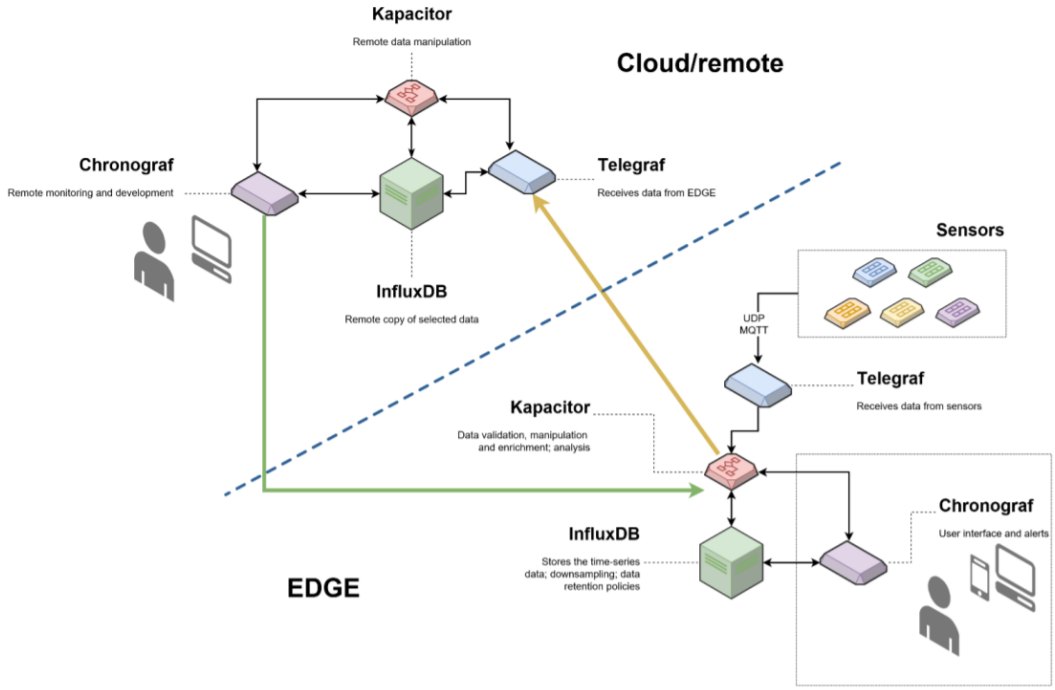
Compose is a tool for defining and running multi-container Docker applications. Using Compose is basically a three-step process[92]. The first one has to define applications environment with a docker-file so it can be reproduced and run anywhere. The second one has to describe the services that Docker will use in docker-compose.yml so they can work together in an isolated environment on the same machine. Lastly, run docker-compose up and compose starts and runs the entire app[92].

6.2.3 InfluxDB

InfluxDB is an open-source time-series database that is part of the TICK (Telegraf, InfluxDB, Chronograf, Kapacitor) stack[90]. See figure 6.2 and figure 6.1. In this project, the generated data was time-series data. The data was analysed with the analytics methods. The database which we have chosen is InfluxDB, as mentioned in the earlier section. InfluxDB is specialised for storing timestamped data which changes over time. The data stored on this database can analysed, forecasting and visualised through plotting. The InfluxDB has features such as the influx stack which consist InfluxDB which is the actual database itself which delivers high performance of writing and storing of time series data. Chronograph which is the user interface and visualisation graph of the time series data which are stored on the InfluxDB and Kapacitor provides alerts and detects anomalies in time series data, and telegraph collects time-series data from all the sources connected to it[89][93].

[94].

Figure 6.1: Grafana

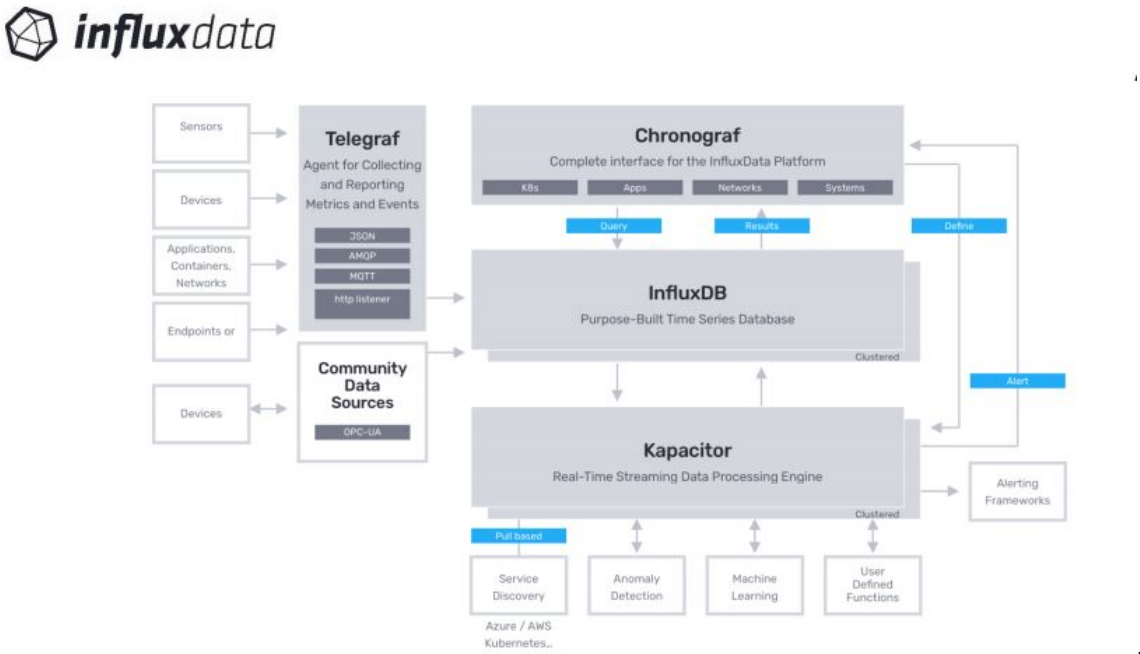


[95].

Figure 6.2: TICK stack overview

6.2.4 TICK Stack

Collectively, Telegraf, InfluxDB, Chronograf and Kapacitor are called the TICK stack. The TICK stack is a loosely linked yet tightly integrated set of open source projects designed to handle massive amounts of timestamped information to support metrics analysis needs. See figure 6.2.



[95].

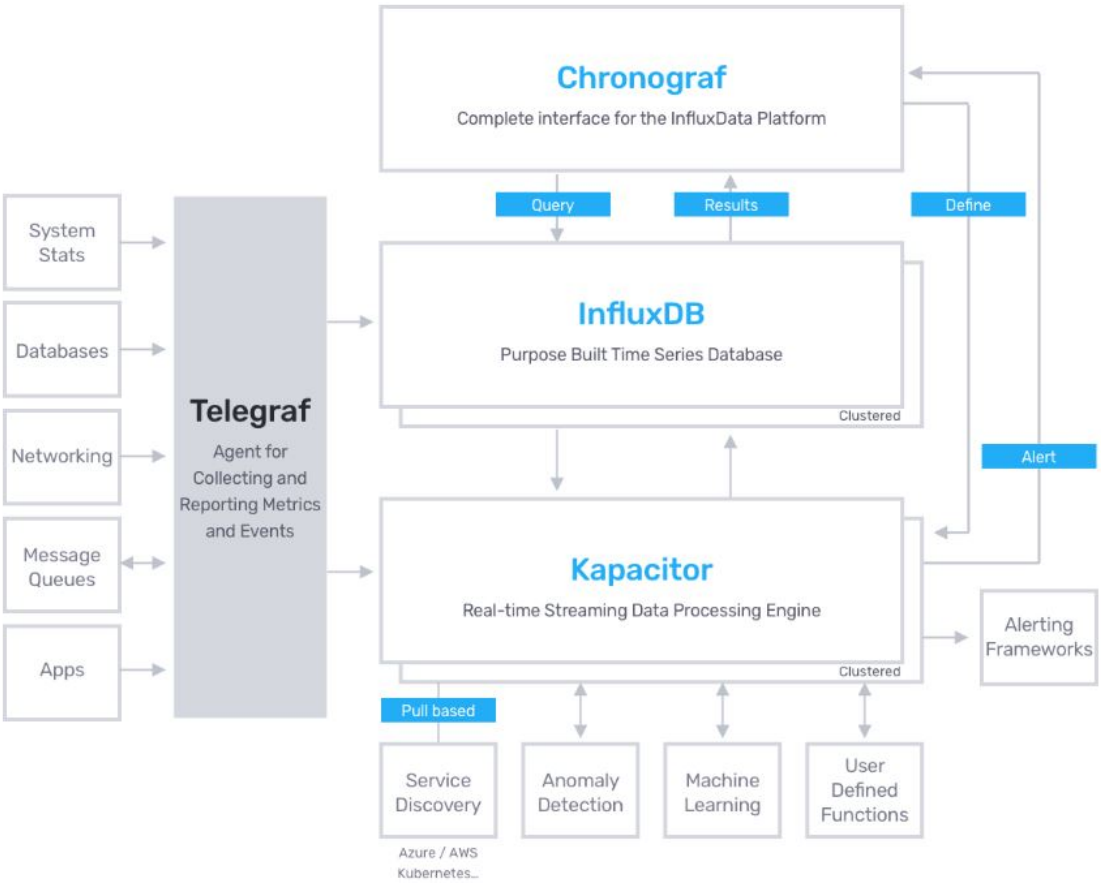
Figure 6.3: TICK Stack overview

6.2.4.1 Telegraf

Telegraf is a plugin-driven server agent for collecting as well as sending metrics and results from databases, systems, and IoT sensors. Figure 6.4 is an example of telegraf. Telegraf connects to data-source's databases and collect data send metrics. Telegraf can collect metrics from a wide of inputs and write them into a wide array of outputs. It is plugin driven for collection and output of data, so it is quickly extendable. [95].

6.2.4.2 Kapacitor

Kapacitor is a native data-processing engine. It processes both stream and batch data from InfluxDB. Kapacitor makes custom logic and user-defined functions for processing alerts with changing thresholds, matches metrics for patterns, computes statistical irregularities as well as performs specific actions based on these alerts. The key features which



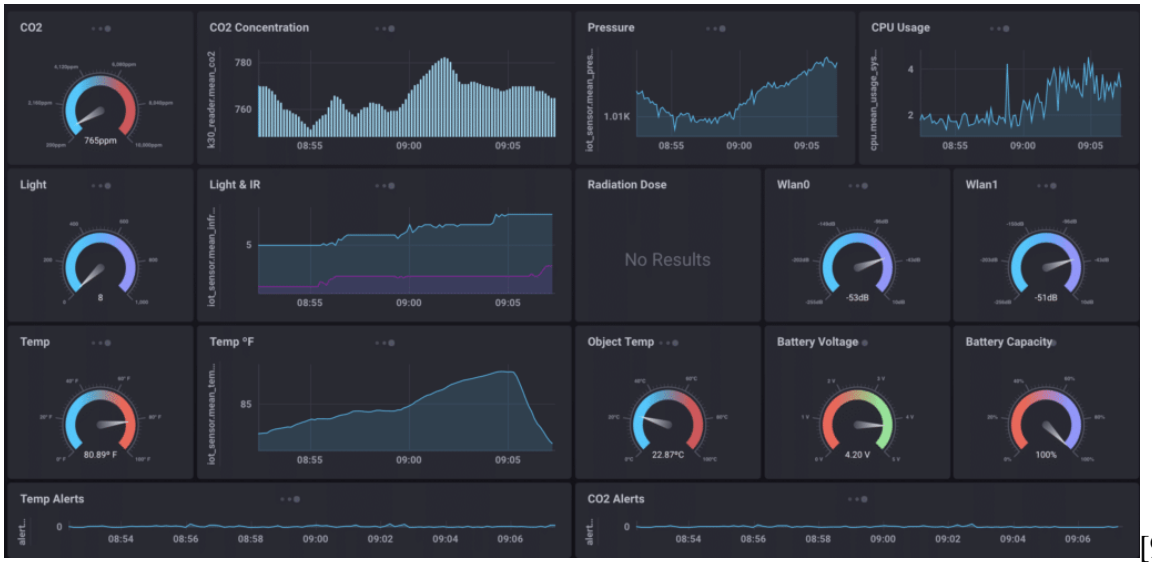
[93].

Figure 6.4: Complete Interface for InfluxData Platform

kapacitor supports are processing streaming batch data, querying data from InfluxDB, performing of data transformation, storing transformed data back into influx DB, and adding custom user-defined functions to detect anomalies[95].

6.2.4.3 Chronograf

Chronograf is the administrative user interface and visualisation engine of the Stack. It makes it easy to set up and maintain the monitoring and alerting for infrastructure. It is easy to use and includes templates and libraries that allow to build dashboards with real-time visualisations of data rapidly and to create alerting and automation rules easily. Chronograf offers a complete dashboard for data visualisation. There are more than twenty dashboards that are available for a quick start. These dashboards can be customised according to the needs of the project. See figure 6.5 as an example of chronograf. Chronograf has security options that one can choose, such as user authentication services. Also, one can create alert services[95].



[93].

Figure 6.5: Example of Chronograph Dashboard

6.3 Data analysis

There might be broken sensors, hence data validation is needed to check if all data from the sensors give the same values. The validation of the data before analysis helps to have accurate results or unbiased results. For example, Vibration data from the capacitive MEMS sensors tend to have the DC bias. It is recommended to filter the data with a high pass filter to clear out the static DC components from the signals. The process

of preparing data for accurate results involves three steps. The first step is to remove accelerometer DC components with high pass filter offset. The second step is to estimate speed through numerical integration of acceleration data, and the third step is to remove speed DC components with high pass filter and so to eliminate the need for an initial speed value[71][96].

6.3.1 Windowing

When Fast Fourier Transform is applied successfully to a periodic signal, and when an integer number of periods is contained in the acquisition time because the FFT is applied to a truncated signal, the problem is the number of signal periods which is not an integer. The acquired signal endpoints are discontinuous, and these discontinuities introduce high frequency and the FFT which might not be the right spectrum of the original signal, but a spread version. The window function is multiplied to the acquired time-domain data set before the FFT. The right window function to use depends on a trade-off between the width of the main lobe, side lobe attenuation, process loss, and spectral leakage. The Hanning window represents the right solution for composite signals typical of rotating equipment. Applying a window function before calculating the FFT helps to manage the spectral leakage. The window function is dependent on the actual signal, but in general, the trade-offs include process loss, spectral leakage, lobe location, and lobe levels[71][96].

6.3.2 Data Analysis with FFT

The Signal processing in vibration analysis starts with the pre-filtering process applied to the accelerometer data. In order to perform signal frequency analysis, vibration time-domain data are discretised and gathered over a time interval and broken down into component frequency waves through an optimised Discrete Fourier Transform (DFT) algorithm which is Fast Fourier TransForm[71][96].

FFT has the following parameters

1. N = discrete time domain samples
2. fs = sampling frequency
3. fmax = fs/2 = maximum frequency spectrum
4. fs/N = frequency resolution

Estimating the spectrum of periodic digital signals which are sampled at a rate of 400Hz with the fundamental period, there are samples from XDK1 and XDK2 in x, y

and z directions Figure 6.6, Figure 6.7, Figure 6.8, Figure 6.9 and Figure 6.10. The frequency spectrum represents the signature of the machine when rotating, and the high vibration that we have not got so far would represent engine imbalance or bearing degradation[71][96].

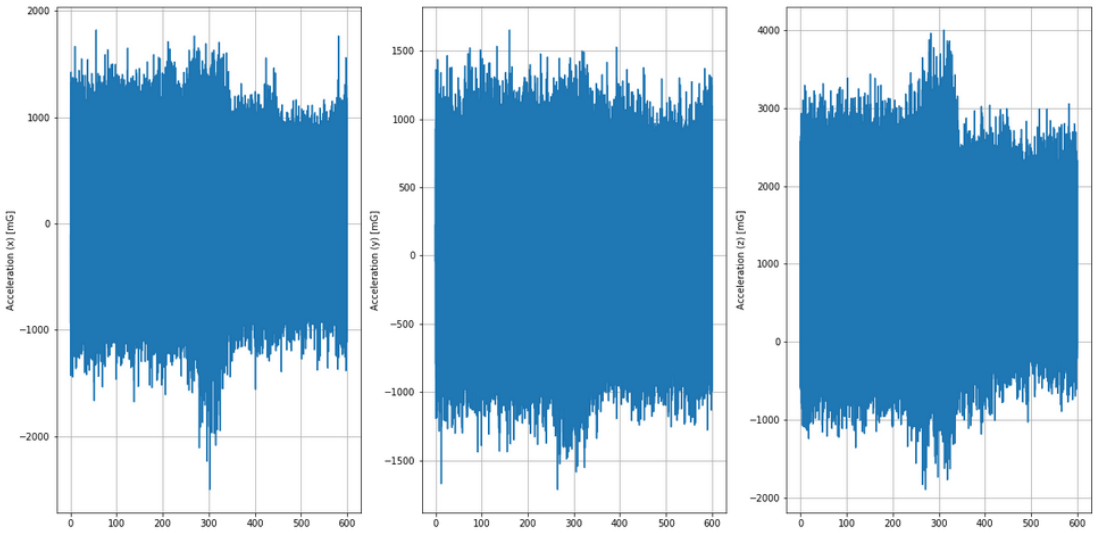


Figure 6.6: Plotting Acceleration XYZ

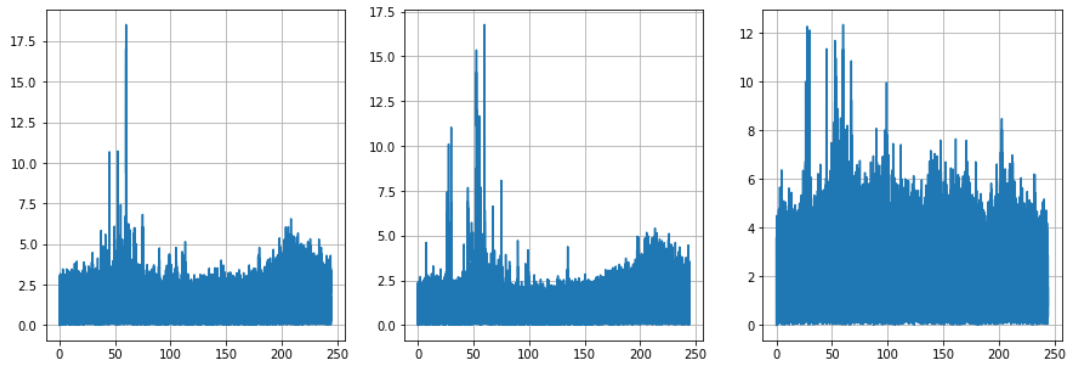


Figure 6.7: FFT

6.3.3 Principal Component Analysis (PCA)

The principal component analysis aims to describe a percentage as large as possible of variation in a set of correlated variables employing the small number of uncorrelated variables. The first principal component has a maximum sample variance among all possible linear combinations. The second principal element is orthogonal to the first, also has the second-largest variance, and so forth. Figure 6.11 shows the frequency of interest in XDK1. The same goes for the second XDK since they are aligned in parallel (see figure

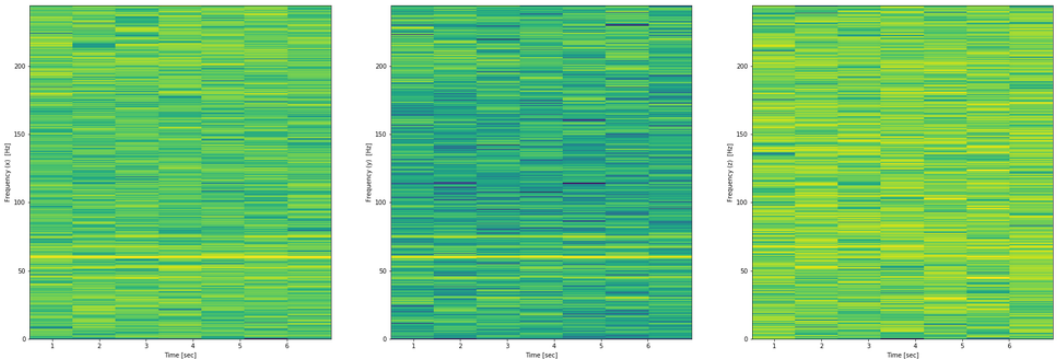


Figure 6.8: Spectrogram XYZ

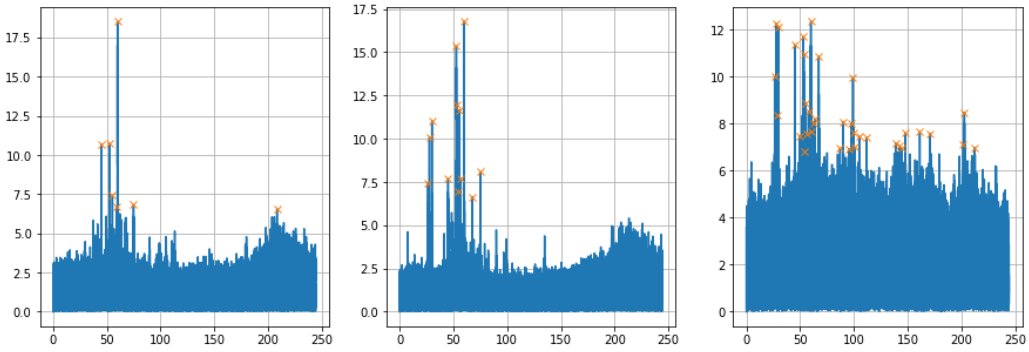


Figure 6.9: Peaks XYZ

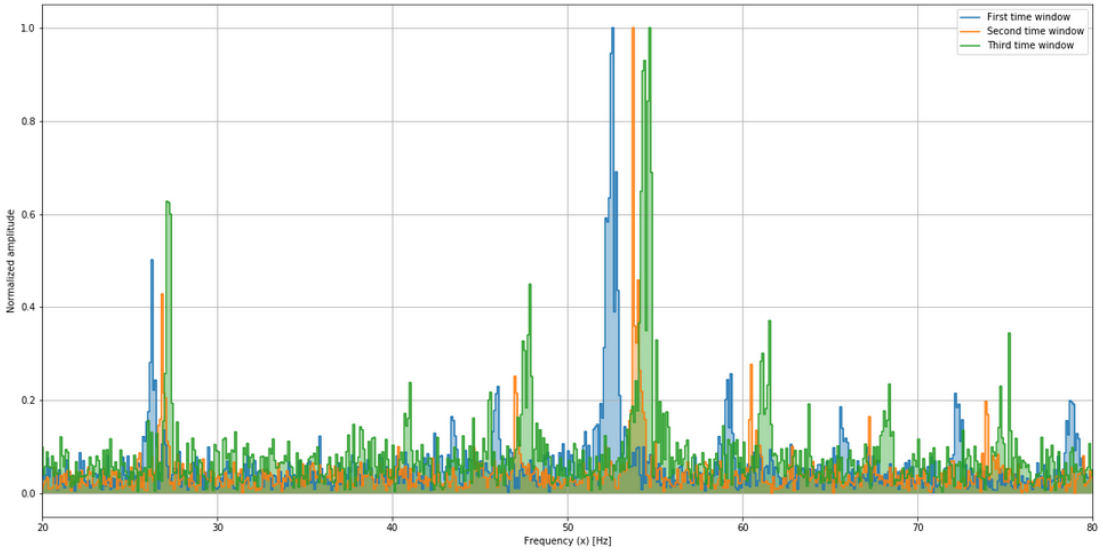


Figure 6.10: Comparison FFT

6.12), so the frequency looks similar to the first one. The plot of the data was from the score vector in the PCA formula of Matlab.

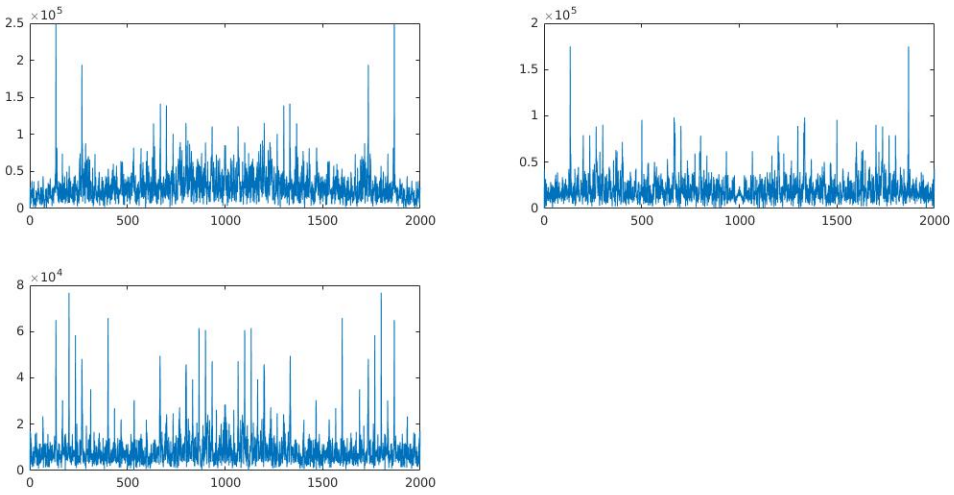


Figure 6.11: PCA XDK1 XYZ

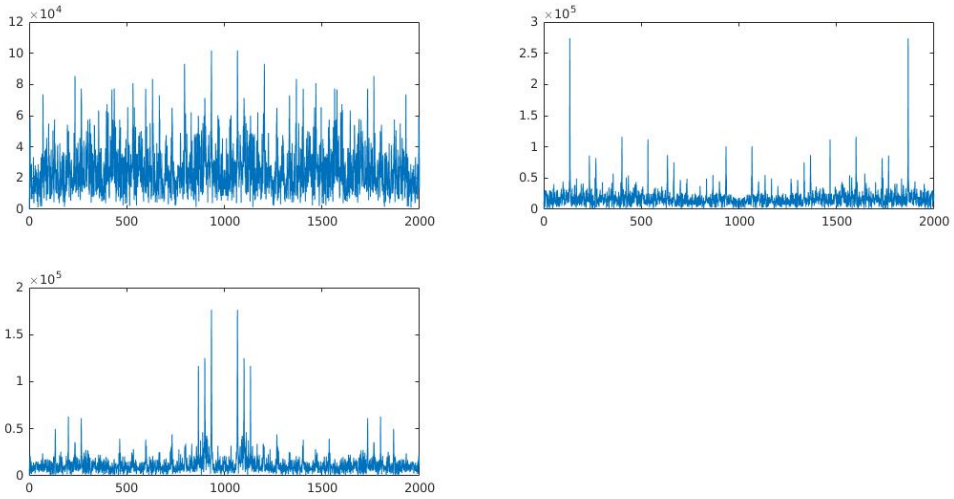


Figure 6.12: PCA XDK2 XYZ

6.3.4 Trend filtering

In the trend filtering method, there were no vivid trends since it is time-series data; the data have much noise; hence one cannot see anything out of the data. In analysing these data from XDK1 and XDK 2 which are time-series data, the data were taken of four

seconds, figure 6.13 shows the frequency and amplitude of the data of XDK1 and figure 6.14 shows the amplitude and frequency of XDK2.

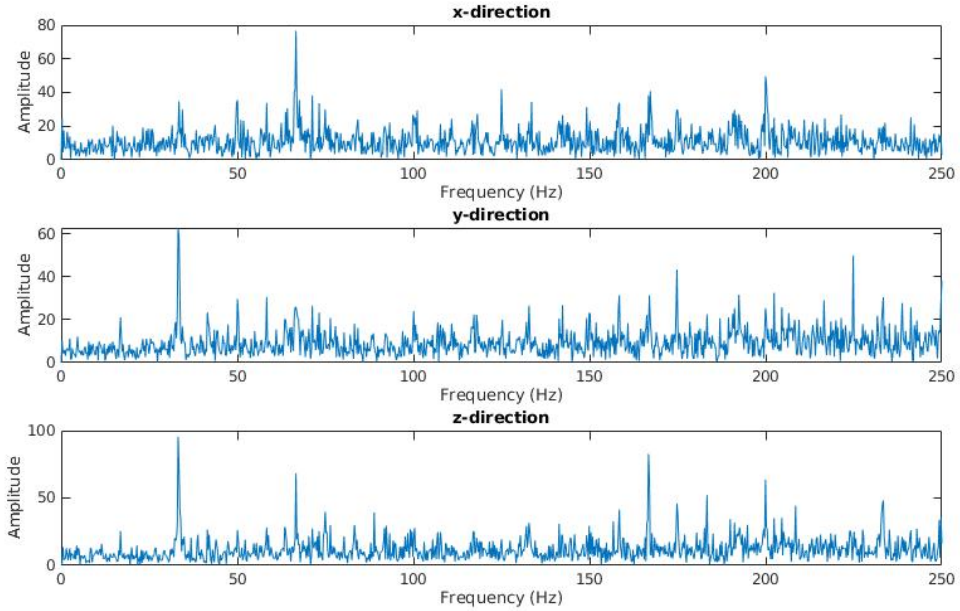


Figure 6.13: Amplitude and Frequency of XDK1 XYZ acceleration

6.3.5 Summary

This chapter explains the data collection and data analysis, and the tools used for data analysis. The chapter explains the acceleration data and how these data are obtained. The software stack is also introduced in this chapter. The software in this project ranges from the database system, software used to obtain data from sensors and software used for data analysis. The chapter also opens up the possibilities of using Docker in the future and how it can be used to deploy the ready products to the field. The database explained in this chapter is Influxdb and the whole TICK Stack. The data analysis methodologies range from the FFT, PCA, Sparse methodologies, or trend filtering methodologies, and these methods show the results from the data we analysed. The following chapter explains the results and evaluation of the project and the quality of the project.

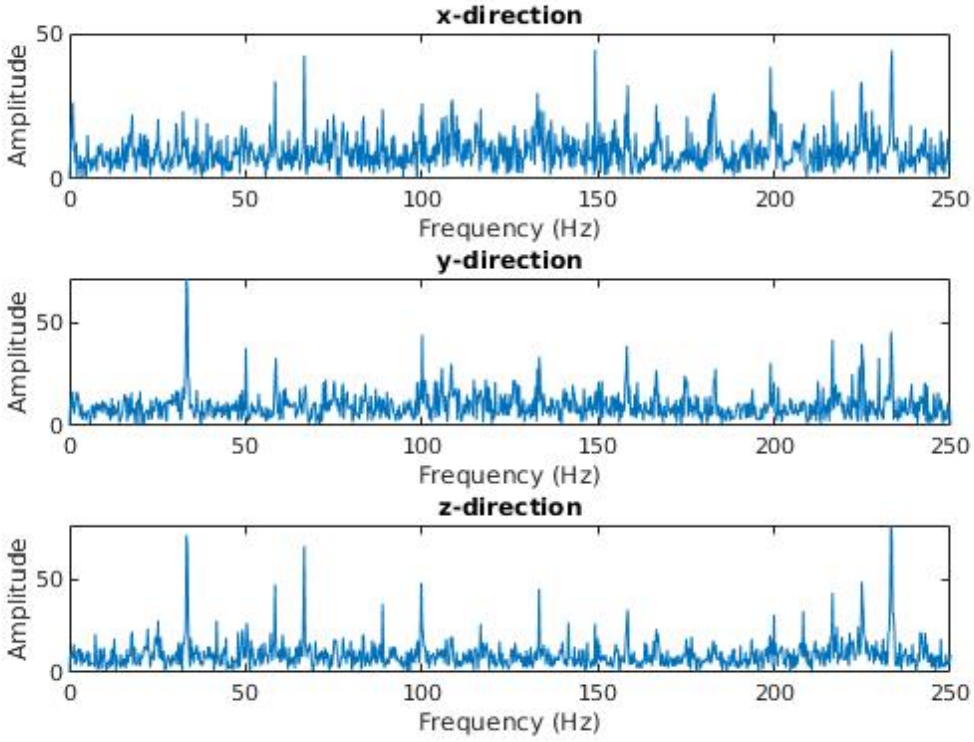


Figure 6.14: Amplitude and Frequency of XDK2 XYZ acceleration

Results and Evaluation

Performance and quality are the main keys to assess the system. predictive maintenance requires the system to be accurate in terms of giving the results of the root cause of the problem and why it should be maintained or fixed. When the system is prone to error, it may result in expensive operation, for example when the system shows that there are bearings that are worn out and hence need changes. The system should be accurate in giving out particular signals according to the previous history of machines or data from similar machines.

7.1 Evaluation Setup

To have clear signals from the sensors in the engine room it is necessary to make sure that there is no other signal interference from other sources such as electric cables. The signals from other sources may be a hindrance to obtaining the real or the intended signal.

7.2 Evaluation Benchmarks

The sensor booster pack was connected to the hardware port of the Raspberry Pi for I2C communication, on pins 3 and 5 (GP-IO 2, GP-IO 3) for SDA and SCL connections. Having the already defined pins for I2C communications was important, as it does not consume much computational power of Raspberry Pi. In the specifications book of a sensor booster pack, it was mentioned that it has to be supplied using a 3V power supply. Several readings of acceleration speed on three-axis on sensor booster pack have been done, recording data at a frequency of 600 Hz. It turned out that making readings at a frequency of 600 Hz was not entirely right, as multiple readings of the same values have been encountered. This issue has been solved by adding a sleep function after each reading. The reading frequency has dropped to 400 Hz and is still on the initial goal requirements of the project.

Evaluation benchmark on InfluxDB, in both cases the Vasa Line and VEBIC case, the time series database is the best so far, as we did not encounter any problems when using

the database. Data were stored in real time. Evaluation benchmark on using wireless sensor vs attached or wired sensor XDK, the advantage of using the XDK is that it is wireless and no cable needs to be attached and can be placed on any location on the testing area. However, the problem is that the battery does not last long when the readings are of higher frequency, for example, more than 500Hz. BoosterPack does not have the problem of the batteries since it is connected to the power cable. The booster pack, however, must be connected to the power cable and also the data cable from the booster pack to the development board which needs to be a long cable.

Evaluation benchmarks on the cables on the wired sensor, for the case of the cables used in the project which are connected from Raspberry pi to the BoosterPack sensor. There was no problem despite the length of the cable which was 30 metres long.

Raspberry Pi is the most widely used development board but it has its shortcomings mostly coming from the SD card failures as we faced a similar problem in this project.

7.3 Summary

In this chapter, we assessed the quality of the project ranging from the tools or hardware stack and also the software stack used in this project. This chapter also assesses the quality of the results from both cases and also the quality of data from both cases. This chapter also suggests the areas where the improvements are needed or where more work can be done in the future. The next chapter chapter will give a summary of the project and give the conclusion and suggest the future work.

Conclusion

This chapter represents the conclusion based on what was done in this thesis. Section 7.1 explains the challenges which seemed to hinder the thesis and the project overall. Subsection 7.1.1 explains the challenges I encountered during this project. Subsection 7.1.2 explains the challenges which others from the previous studies have encountered, and the last subsection suggests the future work of the project.

8.1 Conclusion

This thesis presents the implementation of predictive maintenance PdM using edge computing. The technology does exist in the sense that researchers know the methods of data analysis and how to gather data from sensors. These studies exist, but the vivid, complete project where all the necessary components and technology put together is the missing part. In this project there were challenges, and one of the challenges was how to obtain data with faults. Even though in one occasion the case company tried to make some faults in the engine in order to obtain the data having faults it was not success. Probably in the near future the testers might succeed, but until writing this part, there was no fault data available. The project can be done if the fault data is available. For example, if there is a broken engine and acceleration sensors attached to it, then, the fault data is collected.

The economic aspect that it does not tell so far the return on investment if someone decides to invest in this particular technology.

The sensor readings were precise. We obtained the required readings from the engine when run, but the experienced experts who work in the field suggested more expensive sensors that can handle the ship conditions. The sensors which were used in this project were sensors boosterpack[79],[82].

In the case of real-world factories, from the view of the development cost, it is impracticable to be damaged the expensive target machine deliberately. Actual anomalous sounds occur rarely and have high variability. Exhaustive set of unusual sounds data can not be collected and there is a need to detect unusual sounds for which training data does not exist[17].

Also, according to these researchers [25], there have been several techniques proposed in the literature for feature extraction. It still remains a challenge in implementing a diagnostic tool for real-world monitoring applications, and the reason is because of the complexity of machinery structures and operating conditions.

8.1.1 Future work

There might be the use of GPU in future work. Furthermore, this is due to the power of the processing capabilities of the GPU. There might be the use of docker. That was the reason why we explained it early. The reason for using docker is the ability to transport the application in an isolated environment, which makes it easier for installation.

In future work, it suggested to use a more robust development board to avoid problems of Raspberry Pi, which this project has faced. Also, in the case of using developing the image for the development board, it is suggested to use the yocto project, which is open-source supported by the Linux organisation. The Yocto project is suggested to build it in the host machine, which runs a Linux operating image. In this project, the Yocto project was built on Virtual Machine Vmware Player, which did not support or read the USB. So to avoid such problems, it suggested to run it in a separate machine which has Linux distro such as Ubuntu.

References

[1] J. Wang, Y. Hu, H. Li, and G. Shou, "A lightweight edge computing platform integration video services," in *2018 International Conference on Network Infrastructure and Digital Content (IC-NIDC)*, Aug. 2018, pp. 183–187. DOI: 10.1109/ICNIDC.2018.8525808. (visited on 09/11/2019).

[2] C. C. Byers and P. Wetterwald, "Fog computing distributing data and intelligence for resiliency and scale necessary for iot: The internet of things (ubiquity symposium)," *Ubiquity*, vol. 2015, no. November, 4:1–4:12, Nov. 2015, ISSN: 1530-2180. DOI: 10.1145/2822875. [Online]. Available: http://doi.acm.org/10.1145/2822875 (visited on 09/10/2019).

[3] R. Morabito, V. Cozzolino, A. Y. Ding, N. Beijar, and J. Ott, "Consolidate iot edge computing with lightweight virtualization," *IEEE Network*, vol. 32, no. 1, pp. 102–111, Jan. 2018, ISSN: 0890-8044. DOI: 10.1109/MNET.2018.1700175. (visited on 09/17/2019).

[4] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," in *Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing*, ser. MCC '12, Helsinki, Finland: ACM, 2012, pp. 13–16, ISBN: 978-1-4503-1519-7. DOI: 10.1145/2342509.2342513. [Online]. Available: http://doi.acm.org/10.1145/2342509.2342513.

[5] M. Satyanarayanan, "Pervasive computing: Vision and challenges," *IEEE Personal Communications*, vol. 8, no. 4, pp. 10–17, Aug. 2001, ISSN: 1070-9916. DOI: 10.1109/98.943998. (visited on 09/18/2019).

[6] F. Peng, G. Shou, Y. Hu, and Y. Liu, "Lightweight edge computing network architecture and network performance evaluation," Oct. 2018, pp. 1–3. DOI: 10.1109/ACP.2018.8596139. (visited on 09/11/2019).

[7] S. Yi, Z. Hao, Q. Zhang, Q. Zhang, W. Shi, and Q. Li, "Lavea: Latency-aware video analytics on edge computing platform," in *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*, Jun. 2017, pp. 2573–2574. DOI: 10.1109/ICDCS.2017.182. (visited on 09/11/2019).

[8] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of things (iot): A vision, architectural elements, and future directions," *Future Generation Computer Systems*, vol. 29, no. 7, pp. 1645–1660, 2013, Including Special sections: Cyber-enabled Distributed Computing for Ubiquitous Cloud and Network Services Cloud Computing and Scientific Applications — Big Data, Scalable Analytics, and Beyond, ISSN: 0167-739X. DOI: https://doi.org/10.1016/j.future.2013.01.010. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167739X13000241.

[9] G. Daniel, *2manuscript:machine learning for beginners, machine learning with python*. Amazon kindle version, 2019.

[10] R. Venanzi, B. Kantarci, L. Foschini, and P. Bellavista, "Mqtt-driven node discovery for integrated iot-fog settings revisited: The impact of advertiser dynamicity," in *2018 IEEE Symposium on Service-Oriented System Engineering (SOSE)*, Mar. 2018, pp. 31–39. DOI: 10.1109/SOSE.2018.00013. (visited on 05/19/2019).

[11] I. E. Alguindigue, A. Loskiewicz-Buczak, and R. E. Uhrig, "Monitoring and diagnosis of rolling element bearings using artificial neural networks," *IEEE Transactions on Industrial Electronics*, vol. 40, no. 2, pp. 209–217, Apr. 1993, ISSN: 0278-0046. DOI: 10.1109/41.222642. (visited on 03/16/2019).

[12] P. Wang, P. Tamilselvan, and C. Hu, "Health diagnostics using multi-attribute classification fusion," *Engineering Applications of Artificial Intelligence*, vol. 32, pp. 192–202, 2014, ISSN: 0952-1976. DOI: https://doi.org/10.1016/j.engappai.2014.03.006. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S095219761400061X.

[13] C. McClelland, *What is iot? - a simple explanation of the internet of things*, 2019. [Online]. Available: https://www.iotforall.com/what-is-iot-simple-explanation/ (visited on 07/08/2019).

[14] M. Rouse, *What is internet of things (iot)? - definition from whatis.com*, 2019. [Online]. Available: https://internetofthingsagenda.techtarget.com/definition/Internet-of-Things-IoT (visited on 08/09/2019).

[15] J. Manyika, M. Chui, P. Bisson, J. Woetzel, R. Dobbs, J. Bughin, and D. Aharon, "Unlocking the potential of the internet of things," *McKinsey Global Institute*, 2015. (visited on 01/11/2020).

[16] D. Y. Oh and I. D. Yun, "Residual error based anomaly detection using auto-encoder in smd machine sound," *Sensors*, vol. 18, no. 5, 2018, ISSN: 1424-8220. DOI: 10.3390/s18051308. [Online]. Available: http://www.mdpi.com/1424-8220/18/5/1308.

[17] Y. Koizumi, S. Saito, H. Uematsu, Y. Kawachi, and N. Harada, "Unsupervised detection of anomalous sound based on deep learning and the neyman–pearson lemma," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 27, no. 1, pp. 212–224, Jan. 2019, ISSN: 2329-9290. DOI: 10.1109/TASLP.2018.2877258. (visited on 05/16/2019).

[18] B. Tang, Z. Chen, G. Hefferman, S. Pei, T. Wei, H. He, and Q. Yang, "Incorporating intelligence in fog computing for big data analysis in smart cities," *IEEE Transactions on Industrial Informatics*, vol. 13, no. 5, pp. 2140–2150, Oct. 2017, ISSN: 1551-3203. DOI: 10.1109/TII.2017.2679740.

[19] A. Presbitero, R. Quax, V. Krzhizhanovskaya, and P. Sloot, "Anomaly detection in clinical data of patients undergoing heart surgery," *Procedia Computer Science*, vol. 108, pp. 99–108, 2017, International Conference on Computational Science, ICCS 2017, 12-14 June 2017, Zurich, Switzerland, ISSN: 1877-0509. DOI: https://doi.org/10.1016/j.procs.2017.05.002. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1877050917304805.

[20] P. V. Er and K. K. Tan, "Non-intrusive fall detection monitoring for the elderly based on fuzzy logic," *Measurement*, vol. 124, pp. 91–102, 2018, ISSN: 0263-2241. DOI: https://doi.org/10.1016/j.measurement.2018.04.009. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0263224118302859.

[21] H. Uguz, "A biomedical system based on artificial neural network and principal component analysis for diagnosis of the heart valve diseases," *Journal of Medical Systems*, vol. 36, pp. 61–72, 2010.

[22] M. U. Siddique Latif, J. Qadir, and R. Rana, "Abnormal heartbeat detection using recurrent neural networks," *ArXiv preprint arXiv:1801.08322*, 2018. (visited on 07/10/2019).

[23] M. Delgado, G. Cirrincione, A. Garcıa, J. Ortega, and H. Henao, "Accurate bearing faults classification based on statistical-time features, curvilinear component analysis and neural networks," in *IECON 2012-38th Annual Conference on IEEE Industrial Electronics Society*, IEEE, 2012, pp. 3854–3861. (visited on 03/12/2019).

[24] K. Jafarian, M. Mobin, R. Jafari-Marandi, and E. Rabiei, "Misfire and valve clearance faults detection in the combustion engines based on a multi-sensor vibration signal monitoring," *Measurement*, vol. 128, pp. 527–536, 2018, ISSN: 0263-2241. DOI: https://doi.org/10.1016/j.measurement.2018.04.062. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0263224118303439.

[25] B. Li, P.-l. Zhang, H. Tian, S.-s. Mi, D.-s. Liu, and G.-q. Ren, "A new feature extraction and selection scheme for hybrid fault diagnosis of gearbox," *Expert Systems with Applications*, vol. 38, no. 8, pp. 10 000–10 009, 2011, ISSN: 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2011.02.008. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0957417411002144.

[26] A. Hajnayeb, A. Ghasemloonia, S. Khadem, and M. Moradi, "Application and comparison of an ann-based feature selection method and the genetic algorithm in gearbox fault diagnosis," *Expert Systems with Applications*, vol. 38, no. 8, pp. 10 205–10 209, 2011, ISSN: 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2011.02.065. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0957417411002715.

[27] H. C. Pusey and S. C. Pusey, "Advanced materials and process technology for mechanical failure prevention (proceedings of the meeting of the mechanical failures prevention group (48th) held in wakefield, massachusetts on 19-21 april 1994," VIBRATION INST WILLOWBROOK IL, Tech. Rep., 1994. (visited on 06/11/2019).

[28] C. Li, R.-V. Sanchez, G. Zurita, M. Cerrada, D. Cabrera, and R. E. Vásquez, "Gearbox fault diagnosis based on deep random forest fusion of acoustic and vibratory signals," *Mechanical Systems and Signal Processing*, vol. 76, pp. 283–293, 2016. (visited on 09/11/2019).

[29] H. Oh, M. H. Azarian, and M. Pecht, "Estimation of fan bearing degradation using acoustic emission analysis and mahalanobis distance," in *Proceedings of the Applied Systems Health Management Conference*, 2011, pp. 1–12. (visited on 03/14/2019).

[30] W. Wang and B. Hussin, "Plant residual time modelling based on observed variables in oil samples," *Journal of the Operational Research Society*, vol. 60, no. 6, pp. 789–796, 2009. (visited on 10/11/2019).

[31] "23 - nondestructive inspection and structural health monitoring of aerospace materials," in *Introduction to Aerospace Materials*, A. P. Mouritz, Ed., Woodhead Publishing, 2012, pp. 534–557, ISBN: 978-1-85573-946-8. DOI: https://doi.org/10.1533/9780857095152.534. [Online]. Available: http://www.sciencedirect.com/science/article/pii/B9781855739468500236.

[32] T. Muhammed, R. Mehmood, A. Albeshri, and I. Katib, "Ubehealth: A personalized ubiquitous cloud and edge-enabled networked healthcare system for smart cities," *IEEE Access*, vol. 6, pp. 32 258–32 285, 2018, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2018.2846609.

[33] H. F. Nweke, Y. W. Teh, G. Mujtaba, and M. A. Al-garadi, "Data fusion and multiple classifier systems for human activity detection and health monitoring: Review and open research directions," *Information Fusion*, vol. 46, pp. 147–170, 2019, ISSN: 1566-2535. DOI: https://doi.org/10.1016/j.inffus.2018.06.002. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1566253518304135.

[34] L. Zhang, Z. Zhao, Q. Wu, H. Zhao, H. Xu, and X. Wu, "Energy-aware dynamic resource allocation in uav assisted mobile edge computing over social internet of vehicles," *IEEE Access*, vol. 6, pp. 56 700–56 715, 2018, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2018.2872753.

[35] B. Ashok, S. D. Ashok, and C. R. Kumar, "A review on control system architecture of a si engine management system," *Annual Reviews in Control*, vol. 41, pp. 94–118, 2016, ISSN: 1367-5788. DOI: https://doi.org/10.1016/j.arcontrol.2016.04.005. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1367578816300086.

[36] D. Hall, *Mathematical techniques in multisensor data fusion 2nd ed.* 2004, pp. 31, 94. [Online]. Available: https://bit.ly/2H4HaXj (visited on 05/18/2019).

[37] D. L. Hall and J. Llinas, "An introduction to multisensor data fusion," *Proceedings of the IEEE*, vol. 85, no. 1, pp. 6–23, 1997. (visited on 05/19/2019).

[38] W. Elmenreich, "An introduction to sensor fusion," *Geology*, 2002. [Online]. Available: https://www.researchgate.net/publication/267771481 (visited on 05/16/2019).

[39] E. Todini, "A Bayesian technique for conditioning radar precipitation estimates to rain-gauge measurements," *HYDROLOGY AND EARTH SYSTEM SCIENCES DISCUSSIONS*, vol. 5, no. 2, pp. 187–199, 2001. [Online]. Available: https://hal.archives-ouvertes.fr/hal-00304593.

[40] E. Zervas, A. Mpimpoudis, C. Anagnostopoulos, O. Sekkas, and S. Hadjiefthymiades, "Multisensor data fusion for fire detection," *Information Fusion*, vol. 12, no. 3, pp. 150–159, 2011, Special Issue on Information Fusion in Future Generation Communication Environments, ISSN: 1566-2535. DOI: https://doi.org/10.1016/j.inffus.2009.12.006. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1566253509001006.

[41] J. J. Leonard and H. F. Durrant-Whyte, "Mobile robot localization by tracking geometric beacons," *IEEE Transactions on robotics and Automation*, vol. 7, no. 3, pp. 376–382, 1991.

[42] E. P. Bennett, J. L. Mason, and L. McMillan, "Multispectral bilateral video fusion," *Trans. Img. Proc.*, vol. 16, no. 5, pp. 1185–1194, May 2007, ISSN: 1057-7149. DOI: 10.1109/TIP.2007.894236. [Online]. Available: https://doi.org/10.1109/TIP.2007.894236.

[43] R. Szeliski, *Computer vision: Algorithms and applications*. Springer Science & Business Media, 2010.

[44] O. Sidek and S. Quadri, "A review of data fusion models and systems," *International Journal of Image and Data Fusion*, vol. 3, no. 1, pp. 3–21, 2012.

[45] S. K. Bose, B. Kar, M. Roy, P. K. Gopalakrishnan, and A. Basu, "Adepos: Anomaly detection based power saving for predictive maintenance using edge computing," in *Proceedings of the 24th Asia and South Pacific Design Automation Conference*, ser. ASPDAC '19, New York, NY, USA: ACM, 2019, pp. 597–602. (visited on 06/19/2019).

[46] S. Y. Nikouei, Y. Chen, S. Song, R. Xu, B. Choi, and T. R. Faughnan, "Intelligent surveillance as an edge network service: From harr-cascade, SVM to a lightweight CNN," *CoRR*, vol. abs/1805.00331, 2018. arXiv: 1805.00331. [Online]. Available: http://arxiv.org/abs/1805.00331.

[47] W. Z. Khan, E. Ahmed, S. Hakak, I. Yaqoob, and A. Ahmed, "Edge computing: A survey," *Future Generation Computer Systems*, vol. 97, pp. 219–235, 2019, ISSN: 0167-739X. DOI: https://doi.org/10.1016/j.future.2019.02.050. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167739X18319903.

[48] C. Sonmez, A. Ozgovde, and C. Ersoy, "Edgecloudsim: An environment for performance evaluation of edge computing systems," in *2017 Second International Conference on Fog and Mobile Edge Computing (FMEC)*, May 2017, pp. 39–44. DOI: 10.1109/FMEC.2017.7946405.

[49] D. Zissis, "Intelligent security on the edge of the cloud," in *2017 International Conference on Engineering, Technology and Innovation (ICE/ITMC)*, Jun. 2017, pp. 1066–1070. DOI: 10.1109/ICE.2017.8279999.

[50] S. Dey and A. Mukherjee, "Robotic slam: A review from fog computing and mobile edge computing perspective," in *Adjunct Proceedings of the 13th International Conference on Mobile and Ubiquitous Systems: Computing Networking and Services*, ser. MOBIQUITOUS 2016, Hiroshima, Japan: ACM, 2016, pp. 153–158, ISBN: 978-1-4503-4759-4. DOI: 10.1145/3004010.3004032. [Online]. Available: http://doi.acm.org/10.1145/3004010.3004032.

[51] S. Ci, N. Lin, Y. Zhou, H. Li, and Y. Yang, "A new digital power supply system for fog and edge computing," in *2018 14th International Wireless Communications & Mobile Computing Conference (IWCMC)*, IEEE, 2018, pp. 1513–1517.

[52] H. Truong and M. Karan, "Analytics of performance and data quality for mobile edge cloud applications," in *2018 IEEE 11th International Conference on Cloud Computing (CLOUD)*, Jul. 2018, pp. 660–667. DOI: 10.1109/CLOUD.2018.00091. (visited on 07/19/2019).

[53] A. Minteer, *Analytics for the internet of things (iot)*, 1st ed. Packt Publishing Limited, 2017, pp. 8–24.

[54] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An introduction to statistical learning*.

[55] J. Le, *The 10 statistical techniques data scientists need to master*, 2019. [Online]. Available: https://medium.com/cracking-the-data-science-interview/the-10-statistical-techniques-data-scientists-need-to-master-1ef6dbd531f7 (visited on 09/18/2019).

[56] G. Aurélien, *Hands-on machine learning with scikit-learn and tensorflow*, Second. O'Reily Media, pp. 208–2014. [Online]. Available: www.safaribooksonline.com/library/view/-/9781492032632/?ar. (visited on 01/10/2020).

[57] G. Daniel, *Machine learning with python*. Amazon kindle version, 2019.

[58] w. wiki wiki, *K-means clustering*, 2019. [Online]. Available: https://en.wikipedia.org/wiki/K-means_clustering (visited on 09/11/2019).

[59] L. Tan and J. Jiang, "Chapter 4 - discrete fourier transform and signal spectrum," in *Digital Signal Processing (Third Edition)*, L. Tan and J. Jiang, Eds., Third Edition, Academic Press, 2019, pp. 91–142, ISBN: 978-0-12-815071-9. DOI: https://doi.org/10.1016/B978-0-12-815071-9.00004-X. [Online]. Available: http://www.sciencedirect.com/science/article/pii/B978012815071900004X.

[60] A. Antoniou, *Digital signal processing*. McGraw-Hill, 2006.

[61] M. Manngård and J. M. Böling, "Online frequency estimation with applications to engine and generator sets," *Mechanical Systems and Signal Processing*, vol. 91, pp. 233–249, 2017, ISSN: 0888-3270. DOI: https://doi.org/10.1016/j.ymssp.2016.12.043. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0888327016305465.

[62] M. Chui, J. Manyika, M. Miremadi, N. Henke, R. Chung, P. Nel, and S. Malhotra, *Notes from the ai frontier: Applications and value of deep learning*, 2019. [Online]. Available: https://www.mckinsey.com/featured-insights/artificial-intelligence/notes-from-the-ai-frontier-applications-and-value-of-deep-learning (visited on 09/11/2019).

[63] P. Hillner, *Validation of data from sensor systems*, 2019. [Online]. Available: https://secure.urkund.com/archive/download/52533322-882185-193747 (visited on 09/13/2019).

[64] J. Ravichandran and A. I. Arulappan, "Data validation algorithm for wireless sensor networks," *International Journal of Distributed Sensor Networks*, vol. 9, no. 12, p. 634 278, 2013.

[65] T. Elgamal and M. Hefeeda, "Analysis of pca algorithms in distributed environments," *ArXiv preprint arXiv:1503.05214*, 2015.

[66] J. E. Jackson, *A user's guide to principal components*, 2nd ed. Wiley, 2005.

[67] M. Elad and M. Aharon, "Image denoising via sparse and redundant representations over learned dictionaries," *IEEE Transactions on Image processing*, vol. 15, no. 12, pp. 3736–3745, 2006.

[68] M. Elad, "Sparse and redundant representations - from theory to applications in signal and image processing," 2010.

[69] I. Rish and G. Grabarnik, *Sparse modeling: Theory, algorithms, and applications*. CRC press, 2014.

[70] M. Elad, *Sparse and redundant representations: From theory to applications in signal and image processing*. Springer Science & Business Media, 2010.

[71] M. Looney, *An introduction to mems vibration monitoring | analog devices*, 2020. [Online]. Available: https://www.analog.com/en/analog-dialogue/articles/intro-to-mems-vibration-monitoring.html.

[72] S. Bowers, K. Piety, and R. Piety, "Temporary sensor mounting," in *Proceedingg 4th Incipient Failure Detection Conference*, 1990. (visited on 09/13/2019).

[73] H. Han, S. Cho, and U. Chong, "Fault diagnosis system using lpc coefficients and neural network," in *International Forum on Strategic Technology 2010*, IEEE, 2010, pp. 87–90. (visited on 09/15/2019).

[74] G. Cirrincione, H. Henao, M. Delgado, and J. Ortega, "Bearing fault diagnosis by exin cca," in *The 2012 International Joint Conference on Neural Networks (IJCNN)*, IEEE, 2012, pp. 1–7. (visited on 04/18/2019).

[75] A. Malhi and R. X. Gao, "Pca-based feature selection scheme for machine defect classification," *IEEE Transactions on Instrumentation and Measurement*, vol. 53, no. 6, pp. 1517–1525, Dec. 2004, ISSN: 0018-9456. DOI: 10.1109/TIM.2004.834070. (visited on 12/06/2019).

[76] V. M. Janakiraman and D. Nielsen, "Anomaly detection in aviation data using extreme learning machines," in *2016 International Joint Conference on Neural Networks (IJCNN)*, Jul. 2016, pp. 1993–2000. DOI: 10.1109/IJCNN.2016.7727444. (visited on 07/10/2019).

[77] R. Goodrich, *Accelerometers: What they are how they work*, 2019. [Online]. Available: https://www.livescience.com/40102-accelerometers.html.

[78] D. Dimension, *A beginner's guide to accelerometers*, 2019. [Online]. Available: https://www.dimensionengineering.com/info/accelerometers.

[79] T. Instruments, *Boostxl-sensors sensors boosterpack plug-in module*, 2016. [Online]. Available: http://www.ti.com/lit/ug/slau666b/slau666b.pdf (visited on 03/19/2019).

[80] BOSCH, *Bmi160-datasheet*, 2015. [Online]. Available: https://www.mouser.com/ds/2/783/BST-BMI160-DS000-07-786474.pdf (visited on 03/19/2019).

[81] R. Giometti, *Gnu/linux rapid embedded programming*. 2017.

[82] BOSCH, *Xdk general information guide*, 2017. [Online]. Available: https://bit.ly/2rphM6T (visited on 07/20/2019).

[83] J. Kristoff, *The trouble with udp scanning*, 2019. [Online]. Available: https://condor.depaul.edu/~jkristof/papers/udpscanning.pdf.

[84] W. Wiki, *User datagram protocol*, 2019. [Online]. Available: https://en.wikipedia.org/wiki/User_Datagram_Protocol.

[85] M. MQTT, *Faq - frequently asked questions | mqtt*, 2019. [Online]. Available: http://mqtt.org/faq (visited on 09/20/2019).

[86] A. MXE, *Mxe-5401/m16g adlink technology | mouser*, 2019. [Online]. Available: https://www.mouser.fi/ProductDetail/ADLINK-Technology/MXE-5401-M16G?qs=nwALWJny2wX8OhhQK1pXCw%3D%3D (visited on 07/06/2019).

[87] V. VAASA, *Vebic - vaasa energy business innovation centre*, 2019. [Online]. Available: https://www.univaasa.fi/en/sites/vebic/.

[88] W. Warstila, *Warstila20*, 2019. [Online]. Available: https://www.wartsila.com/docs/default-source/product-files/engines/ms-engine/product-guide-o-e-w20.pdf?utm_source=engines&utm_medium=dieselengines&utm_term=w20&utm_content=productguide&utm_campaign=msleadscoring (visited on 12/11/2019).

[89] M. Rouse, *What is udp (user datagram protocol)? - definition from whatis.com*, 2019. [Online]. Available: https://searchnetworking.techtarget.com/definition/UDP-User-Datagram-Protocol (visited on 10/11/2019).

[90] G. Aasen, *Introduction to influxdata's influxdb and tick stack | influxdata*, 2019. [Online]. Available: https://www.influxdata.com/blog/introduction-to-influxdatas-influxdb-and-tick-stack/ (visited on 06/05/2019).

[91] D. DOCKER, *What is a container? | docker*, 2019. [Online]. Available: https://www.docker.com/resources/what-container (visited on 01/11/2020).

[92] D. Docker-compose, *Overview of docker compose*, 2019. [Online]. Available: https://docs.docker.com/compose/ (visited on 01/11/2020).

[93] H. Mishra, *An introduction to tick stack for iot - iotbyhvm - bits bytes of iot*, 2019. [Online]. Available: https://iotbyhvm.ooo/tick-stack/ (visited on 04/10/2019).

[94] J. LaCruz, *Looking for the perfect dashboard: Influxdb, telegraf and grafana – part xii (native telegraf plugin for vsphere) - the blog of jorge de la cruz*, 2019. [Online]. Available: https://jorgedelacruz.uk/2018/10/01/looking-for-the-perfect-dashboard-influxdb-telegraf-and-grafana-part-xii-native-telegraf-plugin-for-vsphere/ (visited on 09/11/2019).

[95] I. InfluxData, *Telegraf 1.12 documentation | influxdata documentation*, 2019. [Online]. Available: https://docs.influxdata.com/telegraf/v1.12/ (visited on 08/20/2019).

[96] S. B. Chaudhury, M. Sengupta, and K. Mukherjee, "Vibration monitoring of rotating machines using mems accelerometer," *International journal of scientific engineering and research*, vol. 2, no. 9, pp. 5–11, 2014.

Glossary

CPU central processing unit. 29

DFT Discrete Fourier Transform. 51

FFT Fast Fourier Transform. 6, 20, 51

GDN Ground. 29

GPU Graphic Processing Unit. 60

HTTP Hypertext Transfer Protocol. 31

HTTPS Hypertext Transfer Protocol Secure. 31

IoT Internet of Things. 26

JDL Joint Directors of Laboratories. 15

Lo-RA Long Range. 32

MEMS Micro-Electromechanical System. 28, 29, 33

MQTT MQ Telemetry Transport. 32, 33

PCA Principal Component Analysis. 6, 7, 24, 52

PdM predictive maintenance. 4, 59

RPM Revolutions per minute. 37

Soc Socket On Chip. 3

SSD Solid State Drive. 44

TICK Telegraf,InfluxDB,Chronograf,Kapacitor. 48

UDP User Datagram Protocol. 6, 32, 33

USB Universal Serial Bus. 60

WLAN Wire less Local Area Network. 31, 33

Glossary

Layer A collection of related recipes. Layers allow you to consolidate related metadata to customise your build, and isolate information for multiple architecture builds. Layers are hierarchical in their ability to override previous specifications. You can include any number of available layers from the Yocto Project and customise the build by adding your layers after them. The Layer Index is searchable for layers within Yocto Project. 44

OpenEmbedded-Core oe-core is meta-data comprised of foundation recipes, classes and associated files that are meant to be common among many different OpenEmbedded-derived systems, including the Yocto Project. It is a curated subset of an original repository developed by the OpenEmbedded community which has been pared down into a smaller, core set of continuously validated recipes resulting in a tightly controlled and an quality-assured core set of recipes

. 44

Poky is the Yocto Project reference system and is composed of a collection of tools and metadata. It is platform-independent and performs cross-compiling, using the BitBake tool, OpenEmbedded Core, and a default set of metadata. It provides the mechanism to build and combine thousands of distributed open source projects to form a fully customise, complete, and coherent Linux software stack. 44

VEBIC Vaasa Energy Business Innovation Centre or VEBIC is a research and innovation plat-form which is hosted by the University of Vaasa. 37

Wi-Fi wireless network. 31

YOCTO The Yocto Project is an open source collaboration project that provides templates, tools and methods to help you create custom Linux-based systems for embedded products regardless of the hardware architecture. 42

List of Figures

APPENDIX

A.1 Raspberry PI Development Board

The development board was Raspberry Pi 3 and later was changed to Raspberry Pi 4.

A.1.1 Interfacing With I2C Connection

The sensors are connected to the I2C bus on the Raspberry Pi and on the boosterpack. To enable the use I2C from the Raspberry Pi there are configuration which is supposed to be done.

A.2 Source code

The source code used in this project was for the boosterpack sensors, XDK sensors and data analysis.

A.2.1 BoosterPack Source Code

The source code for the BoosterPack which was used in Wasaline line ship.

```

#ifndef _BMI160_ARM_H_
#define _BMI160_ARM_H_

#include <stdint.h>
#include <iostream>
#include <time.h>
#include <stdio.h>
#include <stdint.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include <unistd.h>
#include <linux/i2c-dev.h>
```

```
#include <sys/ioctl.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>
#include "bmi160.h"
#include "bmi160_defs.h"

void delay_ms(uint32_t period);
int8_t i2c_read(uint8_t dev_addr, uint8_t reg_addr, uint8_t *data,
    uint16_t len);
int8_t i2c_write(uint8_t dev_addr, uint8_t reg_addr, uint8_t *data,
     uint16_t len);

struct sensor_data{
  float accx, accy, accz;
};

class BMI160{
  private:
    char *device;
    public:
    BMI160(char *bus_no);
    bmi160_dev initialize(int8_t &rslt);
    void set_sensor_settings(struct bmi160_dev *dev, int mode,
    int8_t &rslt);
    void read_sensor_data(struct bmi160_dev *dev, int8_t &rslt);
    sensor_data read_sensor_struct(struct bmi160_dev *dev, int8_t &
    rslt);
};
#endif
```

Listing A.1: header file

```

#include "bmi160_arm.h"
#include <iostream>
#include <fstream>
#include "influxdb.hpp"

using namespace std;

int main(int argc,  char **argv){

    char *device;
    device = argv[1];
```

```
13
14    BMI160 bmi = BMI160(device);
15
16    int8_t rslt = BMI160_OK;
17  bmi160_dev sensore;
18  sensore = bmi.initialize(rslt);
19    cout << (rslt!=BMI160_OK) << endl;
20  bmi160_dev* p_sensore = &sensore;
21  bmi.set_sensor_settings(p_sensore, 0, rslt);
22    cout << (rslt!=BMI160_OK) << endl;
23
24  influxdb_cpp::server_info si("127.0.0.1", 8094, "db");
25
26  bool rebooted = true;
27  double factor = 8.0;
28
29  while(true){
30      if (rebooted){
31      influxdb_cpp::builder()
32          .meas("reboot")
33      .field("value", "service restarted")
34      //.timestamp(ms)
35      .post_http(si);
36      rebooted = false;
37    }
38      sensor_data results = bmi.read_sensor_struct(p_sensore, rslt)
    ;
39      double accx = double(results.accx)/factor;
40      double accy = double(results.accy)/factor;
41      double accz = double(results.accz)/factor;
42
43      influxdb_cpp::builder()
44      .meas("acceleration")
45    .field("x", accx, 5)
46    .field("y", accy, 5)
47    .field("z", accz, 5)
48    //.timestamp(ms)
49    .send_udp(ipaddress, 8094);
50       //.post_http(si);
51      printf ("X:% .0f \t Y:% .0f \t Z:% .0f \n", accx, accy, accz)
    ;
52      usleep(1000);
53    }
54
55  return 0;
```

```
}
```

Listing A.2: main function

A.2.2 XDK Source Code

The source code for the XDK sensors which was used for the Wärstilä/VEBIC case

```
#include "BCDS_WlanNetworkConfig.h"
#include "Serval_Sntp.h"

#include "em_device.h"
#include "core_cm3.h"
#include <time.h>

#include <inttypes.h>
#include <stdio.h>

#define SNTP_DEFAULT_PORT       UINT16_C(123)
#define SNTP_DEFAULT_ADDR       "0.fi.pool.ntp.org"

uint64_t receivedTimestamp;

static void onTimeReceive(Ip_Address_T* sourceIp, Ip_Port_T
    sourcePort, uint64_t timestamp) {
  receivedTimestamp = timestamp;
}

static void onSent(Msg_T *msg_ptr, retcode_t status) {}

bool setupTime(void) {
    Sntp_initialize();
    Sntp_start(Ip_convertIntToPort(123), onTimeReceive);
    Ip_Address_T destAddr;
    WlanNetworkConfig_GetIpAddress((uint8_t *) SNTP_DEFAULT_ADDR, &
   destAddr);
    Sntp_getTime(&destAddr, Ip_convertIntToPort(SNTP_DEFAULT_PORT),
     onSent);

    return true;
}

uint64_t GetUtcTime()
{
  retcode_t rc = RC_CLOCK_ERROR_FATAL;
  uint64_t sysUpTime;
```

```
  rc = Clock_getTimeMillis(&sysUpTime);
  if (rc != RC_OK)
  {
    printf("Failed to get the Clock Time \r\n");
  }

  return sysUpTime + receivedTimestamp;
}

void getTimestamp(char * buff)
{
  uint64_t timestamp = GetUtcTime();
  sprintf(buff, "%lld", timestamp);
}

bool updateTime(char * buff)
{
  uint64_t oldTimestamp = GetUtcTime();

  uint64_t startTime;
  Clock_getTimeMillis(&startTime);

  bool status = setupTime();

  uint64_t endTime;
  Clock_getTimeMillis(&endTime);

  uint64_t newTimestamp = GetUtcTime();

  //printf("Duration to update time from time server %lld \r\n", (
    endTime-startTime));
  //printf("Drift %lld \r\n", newTimestamp-oldTimestamp-(endTime-
    startTime));
  sprintf(buff, "%lld", newTimestamp-oldTimestamp-(endTime-
    startTime));

  return status;
}

void reboot(void)
{
  NVIC_SystemReset();
}
```

```
78 int checkTime(int hour, int minute)
79 {
80   time_t rawtime = (int64_t)(GetUtcTime()/1000);
81
82   struct tm ts;
83   ts = *localtime(&rawtime);
84
85   //printf("Current time is: %d:%d\n", ts.tm_hour, ts.tm_min);
86   return (hour == ts.tm_hour && minute == ts.tm_min);
87 }
```

Listing A.3: header file

```
1  package main;
2  import platforms.xdk110;
3
4  // External functions
5  native unchecked fn setupTime():        bool header "systime.h";
6  native unchecked fn updateTime(ref: string):  bool header "systime.
       h";
7  native unchecked fn getTimestamp(ref: string):  void header "
       systime.h";
8  native unchecked fn checkTime(h: int32, min: int32):  bool header "
       systime.h";
9  native unchecked fn reboot():   void header "systime.h";
10
11 // Global variables
12 var timeHasBeenObtained: bool = false;    // Whether we have been
       successful in obtaining the time from an SNTP server
13 var tempCorrection = 0.0;         // Correction factor for the XDKs
        heat output
14
15 var lifetime = 0;             // Seconds the device has been alive
16 var lightOn = false;            // If the blinking status lamp (
       yellow) is on
17
18 //
       ------------------------------------------------------------------------
19 //    Device setup
20 //
       ------------------------------------------------------------------------
21
22 setup XDK110
23 {
```

```
  applicationName     =   "bosch_xdk_logger";
  startupDelay      = 5000;
}

setup accelerometer
{
  range          =   Range_4G;
  bandwidth          =   BW_500Hz;
}

setup environment
{
  power_mode         =   Normal;
  standby_time       =   2;
  pressure_oversampling    =   OVERSAMPLE_2X;
}

setup light {}



/*setup gyroscope {}

setup magnetometer {}

setup noise_sensor {}*/

setup led : LED
{
  var red    : bool  = light_up(Red);
  var orange   : bool  = light_up(Orange);
  var yellow   : bool  = light_up(Yellow);
}

setup net: WLAN
{
  ssid         = "EDGE";
  authentication   = Personal(psk = "edgepass");
}

setup mqtt: MQTT
{
  transport = net;
  url = "mqtt://195.148.31.186";
  lastWill = LastWill(topic="XDK3/event", message="Sensor down",
```

```
    qos=0);
  keepAliveInterval = 60;
  clientId = "XDK3";
  var acceleration = topic("XDK3/acc");
  var battery = topic("XDK3/battery");
  var humidity = topic("XDK3/humidity");
  var pressure = topic("XDK3/pressure");
  var temperature = topic("XDK3/temperature");
  var light = topic("XDK3/light");
  var lifetime = topic("XDK3/lifetime");
  var event = topic("XDK3/event");
}

//    ------------------------------------------------------------------------

//    Internal functions
//    ------------------------------------------------------------------------


// Obtains the time from the SNTP server and sets the status light
    (orange)
fn setupAndGetTime()
{
  if (!timeHasBeenObtained)
  {
    led.orange.write(true);
    timeHasBeenObtained = setupTime();

    if (timeHasBeenObtained)
    {
      var timestamp = new string(20);
      getTimestamp(timestamp);

      var message:string = "Time obtained from NTP server.";

      println(`${message}`);
      mqtt.event.write(`${message}`);
    }
    else
    {
      led.orange.write(false);
      println(`Could not obtain time from NTP server.`);
      mqtt.event.write(`Could not obtain time from NTP server.`);
```

```
    }
  }
}

//
    ------------------------------------------------------------------------

//    Event triggers
//
    ------------------------------------------------------------------------


// Is run once on startup of the device
every XDK110.startup
{
  mqtt.event.write(‘Device startup completed.‘);

  // Turn on the red light
  led.red.write(true);
  led.orange.write(false);
  led.yellow.write(false);

  // Gets the time from the SNTP server
  setupAndGetTime();
}

every button_one.pressed {}

every button_two.pressed
{
    mqtt.event.write(‘Performing manual reboot.‘);
    reboot();
}

// Measures the acceleration with 500 Hz
every 2 milliseconds
{
  if (timeHasBeenObtained)
  {
    var timestamp = new string(20);
    getTimestamp(timestamp);

    var accx: double = accelerometer.x_axis.read() * 4 / 8;
    var accy: double = accelerometer.y_axis.read() * 4 / 8;
    var accz: double = accelerometer.z_axis.read() * 4 / 8;
```

```
149     mqtt.acceleration.write(`${timestamp},${accx},${accy},${accz}`)
        ;
150   }
151 }
152
153 // Makes sure we have a time
154 every 1 minute
155 {
156   setupAndGetTime();
157 }
158
159 // Measures the environmental values
160 every 1 minute
161 {
162   var temp : double = environment.temperature.read()/1000.0 -
        tempCorrection;
163     var pres : double = environment.pressure.read()/100.0;
164     var humi = environment.humidity.read();
165     var light = light.intensity.read()/1000.0;
166
167     mqtt.humidity.write(`${humi}`);
168   mqtt.pressure.write(`${pres}`);
169   mqtt.temperature.write(`${temp}`);
170   mqtt.light.write(`${light}`);
171 }
172
173 // Measures the battery status
174 every 2 minute
175 {
176   let ps = XDK110.powerStatus.read();
177     where(ps) {
178         is(PowerStatus.Battery -> level) {
179           if (level > 100.0)
180           {
181             mqtt.battery.write("100.0");
182           }
183           else
184           {
185             mqtt.battery.write(`${level}`);
186           }
187         }
188         is (PowerStatus.Corded)
189         {
190           mqtt.battery.write("100.0");
191         }
```

```
192     }
193 }
194
195 // Updates and sends the lifetime
196 every 1 minute
197 {
198   lifetime+=60;
199   mqtt.lifetime.write(‘${lifetime}‘);
200 }
201
202 // Recalibrates internal clock
203 every 60 minutes
204 {
205   var drift = new string(20);
206   timeHasBeenObtained = updateTime(drift);
207
208   var timestamp = new string(20);
209   getTimestamp(timestamp);
210
211   mqtt.event.write(‘Updated time to: ${timestamp}. Drift: ${drift}
      ms‘);
212 }
213
214 // Blinks the lamp so we know the sensor is alive
215 every 5 seconds
216 {
217   if (lightOn)
218   {
219     led.yellow.write(false);
220     lightOn = false;
221   }
222   else
223   {
224     led.yellow.write(true);
225     lightOn = true;
226   }
227 }
228
229 // Reboots the XDK every night at 2AM (UTC)
230 every 1 minute
231 {
232   if (checkTime(2,00))
233   {
234     mqtt.event.write(‘Performing automated reboot.‘);
235     reboot();
```

```
  }
}
```

Listing A.4: main function

A.2.3 Data Analysis Source Code Matlab

```
clc
clear all,
close all,
Fs = 500;


%acceleration in x-direction
load('xdk_accx.mat')
x=Data;
L=length(x);
n=2^nextpow2(L);
F=fft(x-mean(x),n);
freq = Fs*(0:(n/2))/n;
amp = 2*abs(F/L);

figure,
subplot(3,1,1)
plot(freq,amp(1:n/2+1));
xlabel('Frequency (Hz)')
ylabel('Amplitude')
title('x-direction')

% SDFT
N = 500;              %window length
r = 0.998;           %damping for stability (chosen close to one)
X(1:N/2,1:N) = 0;    %initial Fourier coefficients
psi = x(1:N);
freq_w=linspace(0,Fs/2,N/2);

for n=N+1:length(x)
    psi = [psi(2:end);x(n)]; %update time-window
    for k=1:N/2 %k:th Fourier coefficient
        X(k,n) = X(k,n-1)*r*exp(1i*2*pi*(k-1)/N) - r^N*psi(1) + psi
   (end);
    end

     if mod(n,10)==0
        figure(2), clf
```

```
        plot(freq_w,2*abs(X(:,n))/N)
        title(['Click to continue. n = ', num2str(n)])
         waitforbuttonpress
    end
end

figure(1), hold on , plot(freq_w,2*abs(X(:,end))/N,'r')


%%
%acceleration in y-direction
load('xdk_accy.mat')
y=Datay;
L=length(y);
n=2^nextpow2(L);
F=fft(y-mean(y),n);
freq = Fs*(0:(n/2))/n;
amp = abs(F/L);

subplot(3,1,2)
plot(freq,amp(1:n/2+1));
xlabel('Frequency (Hz)')
ylabel('Amplitude')
title('y-direction')

%acceleration in z-direction
load('xdk_accz.mat')
z=Dataz;
L=length(z);
n=2^nextpow2(L);
F=fft(z-mean(z),n);
freq = Fs*(0:(n/2))/n;
amp = abs(F/L);

subplot(3,1,3)
plot(freq,amp(1:n/2+1));
xlabel('Frequency (Hz)')
ylabel('Amplitude')
title('z-direction')

%acceleration in x-direction
load('xdk_accx2.mat')
x2=Data2x;
L=length(x2);
n=2^nextpow2(L);
```

```
F=fft(x2-mean(x2),n);
freq = Fs*(0:(n/2))/n;
amp = abs(F/L);

figure,
subplot(3,1,1)
plot(freq,amp(1:n/2+1));
xlabel('Frequency (Hz)')
ylabel('Amplitude')
title('x-direction')

%acceleration in y-direction
load('xdk_accy2.mat')
b=Data2y;
L=length(y2);
n=2^nextpow2(L);
F=fft(y2-mean(y2),n);
freq = Fs*(0:(n/2))/n;
amp = abs(F/L);

subplot(3,1,2)
plot(freq,amp(1:n/2+1));
xlabel('Frequency (Hz)')
ylabel('Amplitude')
title('y-direction')

%acceleration in z-direction
load('xdk_accz2.mat')
z2=Data2z;
L=length(z2);
n=2^nextpow2(L);
F=fft(z2-mean(z2),n);
freq = Fs*(0:(n/2))/n;
amp = abs(F/L);

subplot(3,1,3)
plot(freq,amp(1:n/2+1));
xlabel('Frequency (Hz)')
ylabel('Amplitude')
title('z-direction')

%% find the trend x
n = length(y2);
D = zeros(n-2,n);
for k=1:n-2
```

```
128     D(k,k:k+2) = [1 -2 1];
129 end
130
131 lambda=100;
132 %optimization in cvx (l1-trend filtering)
133 tic
134 cvx_begin
135 variable x(n,1)
136     minimize( lambda*norm(D*x2,1) + sum_square(y2-x2) )
137 cvx_end
138 toc
139
140 %optimization in cvx (HP-trend filtering)
141 lambdaHP = 1000;
142 cvx_begin
143 variable xHP(n,1)
144     minimize( lambdaHP*sum_square(D*xHP) + sum_square(y2-xHP) )
145 cvx_end
146
147 %optimization in cvx (elastic net)
148 tic
149 cvx_begin
150 variable x3(n,1)
151     minimize( lambda*sum_square(D*x3) + lambda*norm(D*x3,1) +
    sum_square(y2-x3) )
152 cvx_end
153 toc
154
155 %plot results
156 figure,hold on
157 plot(y2,'k:')
158 plot(x2,'r')
159 plot(xHP,'g')
160 plot(x3,'b')
161 legend('y','l1-trend','HP-trend','Elastic net')
162
163 figure, hold
164 stem(D*x2,'r')
165 stem(D*xHP)
166 stem(D*x3,'b')
167
168  var(y2-x2)
169  var(y2-xHP)
```

Listing A.5: main function

A.3 Yocto Project

The Yocto Project provides open source, high-quality infrastructure and tools to help developers create their own custom Linux distributions for any hardware architecture, across multiple market segments. The Yocto Project is intended to provide a helpful starting point for developers.

The Yocto Project can build tool-chains, boot-loaders, kernels, and root file-systems, also generates an entire Linux distribution for you with binary packages that can be installed at runtime. The Yocto Project is primarily a group of recipes. Yocto is written using a combination of Python and shell script, together with a task scheduler called BitBake that produces whatever you have configured, from the recipes.

A.3.1 Config Files

In the build directory there are two configuration files which are bblayerconfig and local config file which define which type of machine operating system will be built. These files define various configuration variables that govern the project's build process. In general in local conf file is where the image built will be customized.

A.3.1.1 bblayer.config

In this file there are recipes which bitbake will use to build the machine image, these files will be downloaded and used for building the machine.

A.3.1.2 local.config

The local config directory is where the machine image which is going to be built will be defined. The user defined configuration is in this file. For example in this project I2C bus was enabled in this file. The booting system of the machine is defined in this file.